

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Joyce Silva

**Estudo Exploratório e Análise Comparativa de
Ferramentas de Inteligência Artificial
Generativa para o Ensino de Computação**

Uberlândia, Brasil

2024

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Joyce Silva

**Estudo Exploratório e Análise Comparativa de
Ferramentas de Inteligência Artificial Generativa para o
Ensino de Computação**

Trabalho de conclusão de curso apresentado
à Faculdade de Computação da Universidade
Federal de Uberlândia, como parte dos requi-
sitos exigidos para a obtenção título de Ba-
charel em Ciência da Computação.

Orientador: Victor Sobreira, Prof. Dr.

Universidade Federal de Uberlândia – UFU

Faculdade de Computação

Bacharelado em Ciência da Computação

Uberlândia, Brasil

2024

Joyce Silva

Estudo Exploratório e Análise Comparativa de Ferramentas de Inteligência Artificial Generativa para o Ensino de Computação

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, como parte dos requisitos exigidos para a obtenção título de Bacharel em Ciência da Computação.

Victor Sobreira, Prof. Dr.
Orientador

Carlos Cesar Mansur Tuma, Prof. Dr.
Membro da Banca

Carlos Roberto Lopes, Prof. Dr.
Membro da Banca

Uberlândia, Brasil
2024

Resumo

Este trabalho é um estudo exploratório sobre a aplicabilidade de ferramentas de inteligência artificial gerativa no ensino de computação, com foco na análise de respostas a consultas específicas. Foram utilizadas três ferramentas de inteligência artificial gerativa: ChatGPT, Gemini e Copilot. O estudo envolveu a definição e submissão de um conjunto de consultas às ferramentas, com temas típicos da computação, abrangendo conceitos básicos e mais complexos. Diferentes formatos de consulta e funcionalidades das ferramentas foram testados, como a variação no grau de contextualização das consultas e o uso de arquivos. As respostas foram avaliadas com base em sete critérios principais: desvio do assunto, corretude, complexidade, proximidade com a resposta esperada, profundidade, clareza e relevância. Os resultados destacam as melhores respostas de acordo com os tipos de testes realizados, os principais aspectos de cada ferramenta, bem como suas vantagens e limitações de uso. A conclusão aponta que essas ferramentas apresentam diferenças consideráveis de desempenho dependendo do tipo de cenário em que são aplicadas. Com base nisso, a escolha da melhor ferramenta depende do objetivo final desejado. Embora todas suportem a maioria dos recursos, há variação no nível de qualidade das respostas. Este estudo pode auxiliar no processo de escolha da ferramenta mais adequada para diferentes tipos de tarefas que o usuário busca realizar.

Palavras-chave: Inteligência Artificial Generativa, Ensino de Computação, ChatGPT, Gemini, Copilot.

Lista de ilustrações

Figura 1 – Configuração Personalizada do ChatGPT	23
Figura 2 – Figura do ChatGPT Critérios por Tipos de Testes	25
Figura 3 – Resposta de pergunta agrupada no ChatGPT sobre IA	26
Figura 4 – Resposta de pergunta individual no ChatGPT sobre IA	27
Figura 5 – Resposta de pergunta sem contexto no ChatGPT sobre algoritmo	28
Figura 6 – Resposta de pergunta com contexto ChatGPT sobre algoritmo	28
Figura 7 – Figura do Gemini de critérios por tipo de teste	29
Figura 8 – Resposta de pergunta agrupada do Gemini sobre estrutura de dados	30
Figura 9 – Resposta de pergunta individual do Gemini sobre estrutura de dados	30
Figura 10 – Figura do Copilot de critérios por tipo de teste	31
Figura 11 – Resposta de pergunta sem contexto do Copilot sobre estrutura de dados	32
Figura 12 – Resposta de pergunta com contexto do Copilot sobre estrutura de dados	32
Figura 13 – Resposta de pergunta sem contexto do Copilot sobre IA	33
Figura 14 – Resposta de pergunta com contexto do Copilot sobre IA	34
Figura 15 – Figura do ChatGPT de critérios por tipo de teste	35
Figura 16 – Resposta de pergunta agrupadas do ChatGPT sobre mergesort	36
Figura 17 – Resposta de pergunta individual do ChatGPT sobre mergesort	37
Figura 18 – Figura do Gemini de critérios por tipo de teste	37
Figura 19 – resposta de pergunta do Gemini para o teste de variáveis e operações	38
Figura 20 – Figura do Copilot de critérios por tipo de teste	39
Figura 21 – Resposta de pergunta do Copilot para o perguntas de listas parte 1	40
Figura 22 – Resposta de pergunta do Copilot para o perguntas de listas parte 2	40
Figura 23 – Figura de Comparação entre Primeira e Segunda rodada	41
Figura 24 – Resposta de pergunta individual com contexto do ChatGPT sobre algoritmo	42
Figura 25 – Resposta de pergunta no ChatGPT sobre jogo LayerNetwork antes da submissão do PDF	43
Figura 26 – Resposta do ChatGPT sobre jogo LayerNetwork apos submissão de pdf	44
Figura 27 – Resposta do ChatGPT sobre ficha de disciplina com submissão de pdf	45
Figura 28 – Resposta de pergunta do ChatGPT sobre geração de perguntas com submissão de pdf	45
Figura 29 – Resposta de pergunta do ChatGPT sobre questões em pdf	47
Figura 30 – Resposta de pergunta individual com contexto e configuração personalizada do ChatGPT sobre algoritmo	47

Sumário

1	INTRODUÇÃO	7
2	FUNDAMENTAÇÃO TEÓRICA	9
2.1	Inteligência Artificial	9
2.2	Relação entre NLP e IA	10
2.3	O Impacto da GenAI no NLP	10
2.4	Desafios do Ensino de Computação com GenAI	11
2.5	Ferramentas de GenAI	12
2.5.1	ChatGPT	12
2.5.2	Gemini	13
2.5.3	Copilot	13
3	TRABALHOS CORRELATOS	15
3.1	Estudos Exploratórios sobre GenAI	15
3.2	Engenharia de Prompt	16
3.3	Benefícios Potenciais de IA no Ensino	16
4	METODOLOGIA	17
4.1	Processos Iniciais	17
4.2	Critérios	19
4.3	Primeira Rodada: Conceitos Básicos da Computação	20
4.4	Segunda Rodada: Conceitos Específicos de Programação	21
4.5	Testes Especiais	22
5	RESULTADOS E ANÁLISES	24
5.1	Resultados da Primeira Rodada de Testes	24
5.1.1	ChatGPT	25
5.1.2	Gemini	27
5.1.3	Copilot	30
5.2	Resultados da Segunda Rodada de Testes	33
5.2.1	ChatGPT	34
5.2.2	Gemini	35
5.2.3	Copilot	38
5.3	Comparação de Resultados entre Primeira e Segunda Rodada	39
5.4	Resultados de Testes Especiais	41
5.4.1	Testes de Personalização	41

5.4.2	Testes com Arquivos de Dados	42
5.5	Tabelas Comparativas	46
5.5.1	Limitações	48
5.5.2	Ferramentas GenAI e Critérios	48
6	CONCLUSÃO	51
	REFERÊNCIAS	53
	APÊNDICES	57
	APÊNDICE A – PERGUNTAS DA PRIMEIRA E SEGUNDA RODADA DE TESTES	58
A.1	Primeira Rodada	58
A.2	Segunda Rodada	59

1 Introdução

Inteligência Artificial (IA) é uma área que vem se popularizando e ganhando cada vez mais destaque e reconhecimento. Envolve a utilização de algoritmos para solucionar tarefas específicas que, anteriormente, dependiam quase exclusivamente da inteligência humana. Esse crescimento da IA torna a sua exploração ainda mais relevante, pois há grande potencial para facilitar e agilizar o trabalho humano, permitindo-nos concentrar-se apenas no raciocínio crítico e detalhado.

Uma área específica da IA é o processamento de linguagem natural ou NLP (do inglês, *Natural Language Processing*), que vem se destacando mundialmente com a popularização de ferramentas para conversação ou chats. A partir de um prompt, informações do usuário são coletadas e, automaticamente, são geradas respostas rápidas, objetivas e diretas. Uma ferramenta amplamente conhecida é o ChatGPT, desenvolvido e lançado pela OpenAI em 2022 ([WIKIPÉDIA, 2024](#)). Este sistema teve um grande crescimento e vem sendo usado em contextos diversos, como a área acadêmica, inclusive para o ensino.

Essas ferramentas têm grande potencial de aplicação no ensino, por exemplo, as numerosas dúvidas que surgem ao analisar ou estudar uma área específica podem demandar uma quantia exorbitante de tempo de pesquisa em livros ou até na própria internet, utilizando meios como o Google. No entanto, a rapidez com que os chats de IA respondem imediatamente às dúvidas e economizam a maior parte do tempo os torna muito mais atraentes.

Os pesquisadores Kostka e Toncelli destacam o uso de ferramentas de IA no ensino para auxiliar, sem substituir os educadores ([KOSTKA; TONCELLI, 2023](#)). A facilitação no processo de aprendizagem e a dificuldade de aplicação dessas novas tecnologias nas instituições de ensino também são pontos que merecem atenção ([JARA; OCHOA, 2020](#)). Segundo a Educause ([ALEXANDER et al., 2019](#)), 2 a 3 anos são necessários para adoção e adaptação total ao uso de IA no ensino superior.

Outro aspecto a ser levado em consideração é o uso malicioso dessas ferramentas. Devido à facilidade que elas proporcionam, como a geração de textos de linguagem natural, há uma tendência de utilizá-las em trabalhos que deveriam ser feitos inteiramente pelos alunos. Isso torna esse instrumento delicado, principalmente para o desenvolvimento dos alunos, uma vez que eles podem se tornar dependentes e deixar de desenvolver seu próprio pensamento analítico, assim como suas capacidades de escrever e elaborar, por exemplo, uma dissertação.

Este trabalho explora algumas das vantagens e desvantagens no uso de ferramentas populares de IA Generativa ou GenAI (do inglês, Generative AI) para o Ensino de

Computação. Ele destaca possíveis aplicações, evoluções e desenvolvimentos, incluindo a integração com conceitos de engenharia de prompt para reforçar e explorar o conteúdo. Inserido no campo das tecnologias para o Ensino de Computação, ele visa contribuir com o processo de ensino e aprendizagem. Aos envolvidos na área, eles poderão utilizar o conteúdo deste estudo para obter diretrizes sobre as melhores funções das ferramentas de IA, como elas respondem e como podem catalisar processos que, de outra forma, poderiam demandar mais tempo e esforço.

Esse direcionamento foi realizado por meio de diversos testes em três ferramentas, nos quais foram registradas suas respostas. Ao final, foi feita uma comparação para identificar o que funcionou melhor em cada uma, possibilitando recomendar ao usuário as melhores formas de utilizar essas ferramentas. Todos os testes foram realizados priorizando o uso de planos gratuitos, a fim de melhorar a acessibilidade e usabilidade para alunos de programação e demais interessados. Com base nas análises e comparativos realizados, propomos algumas diretrizes para utilização dessas ferramentas, em especial, no contexto de ensino da computação.

2 Fundamentação Teórica

Ao longo deste capítulo, exploraremos desde os fundamentos da IA até suas aplicações específicas no ensino de programação, com enfoque especial na interseção entre *Generative artificial intelligence* (GenAI) e *Natural Language Processing* (NLP). Na seção 2.1 mostra o conceito e exploração da IA. Na seção 2.2 a relação entre NLP e IA. Já na 2.3 o impacto da GenAI no NLP. Na seção 2.4 os desafios do ensino de computação com GenAI e por fim a seção 2.5 com as ferramentas que serão abordadas neste estudo.

2.1 Inteligência Artificial

Uma definição clássica de IA seria: 'Inteligência Artificial é o estudo de agentes inteligentes, ou seja, sistemas que percebem seu ambiente e tomam ações que maximizam suas chances de alcançar um objetivo específico' ([RUSSELL; NORVIG, 2016a](#)). Com ela, é possível ampliar horizontes e ter uma maior gama de conteúdo para ser utilizado em diversas situações. Pouco se explica para a compreensão da grande massa da população, que imagina a IA como algo que se cria e funciona por si só, como se tivesse vida própria. No entanto, poucos entendem que são algoritmos criados por humanos para executar e facilitar nossas tarefas e que, ao contrário da mente humana, a IA é limitada em sua capacidade de raciocínio lógico e processamento emocional.

A IA está cada vez mais presente em nossa vida diária: no trabalho, nas indústrias, no entretenimento, saúde e no ensino. Isso ressalta a importância de abordar essa tecnologia com mais profundidade. Apesar de parecer inovadora e recente, a IA não é tão nova quanto parece. Desde 1943, com o primeiro modelo de rede neural proposto pelos neurofisiologistas W. McCulloch e W. Pitts, ela vem ganhando destaque. Portanto, seus fundamentos têm origem em muitos anos atrás. No entanto, naquela época, os recursos tecnológicos não eram suficientes para avançar, nem os investimentos eram favoráveis. O desenvolvimento da IA, devido a essas limitações, passou por diversos períodos de estagnação, conhecidos como 'invernos' da IA. Durante esses períodos, não era possível conduzir pesquisas significativas nessa área devido à falta de investimento e valorização ([JIANG et al., 2022](#)). Com o tempo, entretanto, o potencial da IA para transformar o mundo em que vivemos foi reconhecido, e gradualmente ela passou a desempenhar um papel mais significativo no cenário atual.

A IA é um campo muito amplo que inclui o desenvolvimento de sistemas e algoritmos que são capazes de fazer tarefas que normalmente precisaria da inteligência humana ([RUSSELL, 2022](#)). Podem ser utilizadas várias funções em seu desenvolvimento, tais como redes neurais, algoritmos genéticos e lógica difusa, sendo construída através de

alguns passos fundamentais: coleta de dados, que basicamente é a fonte de alimentação para a IA, sendo imagens, textos, números, etc.; processamento dos dados, onde os dados são tratados e normalizados; aprendizado, onde o algoritmo é treinado com base em experiências, utilizando aprendizado supervisionado ou não supervisionado, e 'aprende' o que é certo e errado com base nos dados apresentados e no tipo de treinamento; feedback e melhoria contínua, onde, com os resultados obtidos, o modelo é melhorado e ajustado para aprimorar o desempenho e performance ([RUSSELL; NORVIG, 2016b](#)).

2.2 Relação entre NLP e IA

O NLP é um campo dentro da IA que explora a interação entre a linguagem humana e a linguagem computacional. Sua principal função é permitir que sistemas de computadores interpretem a linguagem humana e vice-versa, além de gerar respostas que sejam o mais próximas possível da linguagem humana de forma eficiente ([GO; SUNDAR, 2019](#)).

Nela, existem alguns procedimentos para o seu funcionamento, como a análise de texto, que consiste em analisar a escrita, dividindo-a em pedaços menores, chamado de tokenização, sendo possível a identificação de sujeitos, predicado, objeto, etc. A análise sintática também é outro ponto importante, pois nela fica claro o entendimento da estrutura gramatical, verificando a organização das frases. A análise semântica, por sua vez, já analisa o sentido da construção das palavras e o significado das frases. A desambiguação é essencial, pois algumas sentenças podem ter significado ambíguo; essa etapa cuida disso com o apoio de dicionários e outros recursos. Quanto à geração de texto, assim como o sistema precisa interpretar a linguagem, ele também pode precisar gerar frases. Nesse contexto, o mesmo esquema é seguido para gerar frases com sentido semântico e sintático, o mais próximo possível do real. ([MARTIN, 2009](#))

As aplicações são muitas, como em assistentes virtuais, corretores ortográficos, sistemas de busca e análise de sentimentos em mídias sociais ([KOROTEEV, 2021](#)). Há inúmeras funcionalidades, e muitas vezes não é de conhecimento geral que essa tecnologia é utilizada em diversos contextos, por serem tão naturais e intrínsecas. No entanto, ela está presente em várias áreas.

2.3 O Impacto da GenAI no NLP

A IA teve uma evolução notável, em grande parte pelos avanços tecnológicos e computacionais, que viabilizaram a aplicação de modelos e algoritmos previamente desenvolvidos. A melhoria nas formas de interação e no processamento de linguagem natural é um desses destaques, com a popularização dos chatbots ([SCHACHNER; KELLER;](#)

WANGENHEIM, 2020). Essas ferramentas são definidas como GenAI, tendo como principal função gerar novos conteúdos, tomando como base a grande quantidade de dados contidos e processados por elas, a partir do parâmetro de entrada fornecido pelo usuário. (GUPTA, 2024)

No contexto da linguagem natural, as GenAI atuam como geradoras de textos coerentes, capazes de responder perguntas, criar música, poesia, dissertações, entre outros, de forma semelhante aos humanos. Isso traz facilidade no processo de criação e desenvolvimento de ideias, tarefas e conceitos. Ferramentas como o ChatGPT 2.5.1, Gemini 2.5.2 e Copilot 2.5.3 se destacaram nesse contexto.

2.4 Desafios do Ensino de Computação com GenAI

No campo da programação, tanto professores quanto alunos podem enfrentar diversos desafios, que podem ser tanto positivos quanto negativos. Muitos desses desafios decorrem da complexidade inerente ao ensino em áreas relacionadas à tecnologia.

A programação envolve a compreensão de conceitos bem complexos, como lógica de programação, estruturas de dados, algoritmos, e paradigmas de programação. Os estudantes têm algumas dificuldades em entender e aplicar esses conceitos, o que pode levar ao desânimo (WANG; LIU, 2020). Essa complexidade pede uma abordagem que facilite o entendimento na prática. Sem esse suporte, muitos estudantes podem acabar desistindo ou não indo tão bem (ROBINS; ROUNTREE; ROUNTREE, 2003). A utilização de ferramentas de IA no ensino de programação pode atuar como uma ajuda extra, essas facilidades podem fornecer explicações detalhadas e respostas rápidas a perguntas, ajudando os alunos a superar problemas.

Um dos desafios da programação é a personalização do ensino, que é um aspecto fundamental para entender as individualidades de cada aluno. Os métodos de ensino tradicional adotam uma abordagem padrão com metodologias mais engessadas, o que dificulta a compreensão do conteúdo em muitos momentos (SMITH; JONES, 2019). A IA nesse ramo pode trazer grandes facilidades, usufruindo do progresso dos alunos para adaptar o conteúdo. Sistemas tutores inteligentes, por exemplo, podem auxiliar e dar um feedback instantâneo para o aluno e orientações fora do horário institucional, os deixando mais satisfeitos na aprendizagem (BROWN; GREEN, 2021; LEE; KIM, 2020).

Outro desafio no ensino de programação é manter os alunos engajados e motivados. Eles possuem diversos interesses e manter o foco total no desempenho acadêmico pode ser um obstáculo a superar. (DAVIS; FREDERICKS, 2017). As ferramentas de IA podem ajudar no engajamento desses alunos, por exemplo, por meio do conceito de gamificação, com jogos de aprendizagem que podem transformar a aprendizagem mais divertida e dinâmica, fazendo os alunos a participaremativamente (LEE; KIM, 2020).

2.5 Ferramentas de GenAI

Existem cada vez mais ferramentas relacionadas à GenAI, muitas das quais se destacam por sua desenvoltura excepcional, trazendo grande valor para os usuários. Neste estudo, serão apresentadas três ferramentas de GenAI, com cada seção dedicada à conceituação de uma delas. Na seção 2.5.1, são fornecidas informações sobre o ChatGPT, na 2.5.2 sobre o Gemini e, por fim, na 2.5.3, sobre o Copilot.

2.5.1 ChatGPT

O ChatGPT é uma ferramenta de IA desenvolvida e lançada pela OpenAI em 2022 ([WIKIPÉDIA, 2024](#)). Este sistema teve um grande crescimento e se popularizou principalmente nas áreas da educação e acadêmica. Ele usa o NLP para gerar respostas semelhantes às humanas, sendo fundamentado no modelo Generative Pre-trained Transformer (GPT) ([AN; DING; LIN, 2023](#)).

O termo 'Generativo' refere-se à capacidade de gerar texto. Em vez de apenas entender ou classificar texto, essa tecnologia pode criar novas frases, parágrafos ou até mesmo artigos completos a partir de prompts ou instruções dadas ([BROWN et al., 2020](#)). O termo 'Pré-treinado' significa que o modelo é inicialmente treinado com uma grande quantidade de dados textuais, como livros, artigos e websites ([DEVLIN et al., 2019](#)). Esse treinamento inicial é feito de forma não supervisionada, o que significa que o modelo aprende apenas observando os dados, sem precisar de rótulos ou categorizações específicas ([RADFORD et al., 2018](#)). Já o termo 'Transformer' é uma classe de modelos de aprendizado profundo introduzida em 2017 por Vaswani et al., que trouxe uma nova capacidade ao NLP de processar grandes quantidades de texto de forma mais eficiente e eficaz em comparação com modelos anteriores ([VASWANI et al., 2023](#)).

Quando o usuário insere algum dado, algoritmos de aprendizado de máquina são aplicados a uma grande quantidade de texto para gerar uma resposta semelhante à humana. A arquitetura é baseada em redes neurais de transformação, onde o modelo é treinado com dados e aprende a prever a próxima palavra com base nas anteriores ([AN; DING; LIN, 2023](#)).

O plano gratuito do ChatGPT tem algumas limitações em relação ao uso e recursos. Durante horários de pico, pode ter restrições de acesso, com usuários assinados tendo prioridade. A quantidade de tokens (ou caracteres) por interação também é limitada, o que pode restringir o tamanho das respostas e das mensagens enviadas. Recursos extras, como versões mais avançadas do modelo, plugins e outras funcionalidades, não estão disponíveis no plano gratuito. ([OPENAI, 2024a; OPENAI, 2024b](#)).

O ChatGPT foi escolhido porque é uma das ferramentas de IA mais conhecidas e usadas na atualidade, com fortes indicações e recomendações gerais. A partir de sua

definição, poderemos compreender melhor as comparações entre as diferentes ferramentas analisadas.

2.5.2 Gemini

Gemini é uma ferramenta de IA multimodal, ele tem a capacidade de manipular diversos tipos diferentes de dados, que podem incluir textos, áudios e imagens. Ele é bem recente, criada em 2023 pela Google DeepMind, tem evoluído bastante conforme vai sendo popularizada entre a comunidade ([IMRAN; ALMUSHARRAF, 2024](#)).

A tecnologia utilizada pelo Gemini IA é a GenAI multimodal, que emprega redes neurais profundas para aprender padrões complexos a partir de grandes conjuntos de dados. Essa tecnologia possui a capacidade de gerar novos conteúdos, incluindo a criação de textos, imagens, músicas e muito mais ([CHO et al., 2014](#)).

O Gemini pode ser usado em alguns produtos do Google, como o Google Search, para melhorar a publicidade direcionada e dar recomendações personalizadas. Com a capacidade de lidar com diferentes tipos de dados, como texto e imagens, o Gemini pode tornar a experiência do usuário mais interessante e envolvente ([SAEIDNIA, 2023](#)).

Na educação, o Gemini pode atuar como um assistente virtual multimodal, ajudando estudantes e pesquisadores a encontrar rapidamente informações relevantes. Ele pode fornecer explicações detalhadas, sugerir recursos complementares e auxiliar na resolução de problemas complexos, utilizando texto, gráficos e vídeos para tornar o conhecimento mais acessível e facilitar o aprendizado de forma interativa e dinâmica ([MEKNI, 2021](#)).

O plano gratuito do Gemini tem algumas limitações que variam, mas normalmente inclui um limite de tokens, as unidades de texto que processa por mês. Algumas funcionalidades avançadas, como recursos extras, ficam disponíveis apenas para assinantes, como os Gem, que é um recurso personalizados para o chat. Outra restrição é que certos tipos de conteúdo, como arquivos PDF, não podem ser submetidos no plano gratuito ([GOOGLE, 2024b](#); [GOOGLE, 2024a](#); [GOOGLE, 2024c](#)).

2.5.3 Copilot

O Copilot é uma ferramenta inovadora de IA desenvolvida pela Microsoft. Ele é composto por 2 modelos de linguagem da OpenAI, o GPT e o DALL E. Ele mescla sua solução a diversas das suas próprias ferramentas como Windows 11, Bing Search, Microsoft Edge e Microsoft 365 ([REBELO, 2023](#)).

Um dos grandes destaques do Microsoft Copilot é sua habilidade de fornecer respostas apropriadas em tempo real durante as interações dos usuários. Ao compreender o

contexto e a intenção da conversa, o Copilot agiliza a comunicação, tornando as conversas mais cativantes e facilitando para os usuários ([ADETAYO; ABORISADE; SANNI, 2024](#)).

O Microsoft Copilot oferece três estilos de conversação para se adequar às diferentes necessidades dos usuários: mais preciso, mais equilibrado e mais criativo. No modo 'mais preciso', o Copilot adota um tom profissional e direto, com informações detalhadas, claras e estruturadas. O modo 'mais equilibrado' apresenta um jeito mais amigável, combinando fatos científicos com curiosidades interessantes. Já no modo 'mais criativo', o Copilot é mais imaginativo e expressivo, tendo informações básicas, junto com opiniões pessoais e algumas sugestões. Essa flexibilidade permite que os usuários escolham o estilo de resposta que melhor se adapta ao seu desejo final ([BATT, 2023](#)).

A versão gratuita do Copilot tem algumas limitações. Ela só pode ser acessada via web, enquanto a versão paga tem integração com aplicativos do Microsoft 365. Embora tenha funções essenciais, a versão gratuita não inclui as capacidades mais avançadas da versão paga. Para criar apresentações no PowerPoint, por exemplo, é necessário transferir manualmente o conteúdo gerado. Também tem um limite por dia de 30 imagens por usuário. Também atualmente não suporta upload de arquivos diferente de imagens ([MICROSOFT, 2024b](#); [MICROSOFT, 2024a](#); [MICROSOFT, 2024c](#)).

3 Trabalhos Correlatos

Este capítulo apresenta alguns dos trabalhos relacionados aos temas abordados nessa monografia, em especial, considerando o contexto de aplicação de IA à educação. Os trabalhos da seção 3.1 por exemplo, focam no estudo exploratórios de IA. Já o trabalho da seção 3.2 apresenta conceitos de engenharia de prompt. Além disso, os trabalhos do 3.3 oferecem pontos valiosos sobre os benefícios potenciais de IA no ensino.

3.1 Estudos Exploratórios sobre GenAI

O artigo de [Cooper \(2023\)](#) aborda a utilização e exploração do ChatGPT na educação científica, considerando as respostas a perguntas sobre ensino e como os professores podem usar essa ferramenta para facilitar e agilizar processos em suas pesquisas. Os resultados mostram que o ChatGPT alinha suas respostas a temas relevantes da literatura acadêmica, como o ensino baseado em investigação e a importância de conectar o aprendizado ao mundo real. No entanto, o estudo destaca algumas limitações, como a falta de evidências que comprovem as respostas, evidenciando um ponto crítico das IA: o risco de suas respostas serem consideradas verdadeiras sem referências concretas. Problemas éticos, como viés nos dados e impactos ambientais, também foram mencionados. Os professores podem usar o ChatGPT para criar planos de aula, questionários e rubricas, mas é fundamental que adaptem e critiquem esses materiais para que se adequem aos seus contextos específicos. Semelhante ao estudo exploratório mencionado, esta monografia analisa o ChatGPT, o Gemini e o Copilot como ferramentas de GenAI, com foco em perguntas e análises na área de computação.

O artigo [Leite \(2024\)](#) é focado na comparação das três ferramenta de chatbots, ChatGPT, Gemini e Copilot, em como elas podem ser usadas no ensino de química, analisando as repostas e o quanto precisas elas são, para os conceitos básicos (átomo, elétron, mol, molécula e substância química). Os resultados indicam que, embora as respostas sejam coerentes e auxiliem no aprendizado, elas diferem em complexidade e nível de detalhamento. O estudo trouxe um exemplo do Gemini que apresentou respostas mais detalhadas, incluindo analogias e contexto histórico, enquanto o ChatGPT ofereceu definições básicas. O Copilot no artigo foi o único a citar as fontes. A pesquisa destaca os riscos e cautela que se deve ter ao usar IA na educação, pois podem trazer respostas incompletas ou superficiais. Ao final, concluiu que, embora essas ferramentas possam acrescentar e complementar o ensino, professores e alunos devem usá-las de forma crítica, verificando a precisão e levando em conta o uso ético e a limitação do raciocínio humano. Este estudo tem um conteúdo muito parecido com esta monografia, ambos utilizaram de comparações entre

as três ferramentas de GenAI, o ChatGPT , Gemini e Copilot, levando em comparação o ensino acadêmico, porém no artigo citado é voltado para o ensino de química, enquanto neste estudo se aprofunda sobre a computação.

3.2 Engenharia de Prompt

O artigo [Chen et al. \(2023\)](#) informa e discute o uso e a importância da engenharia de prompt nas submissões de ferramentas de GenAI. O uso de prompts eficazes é destacado como uma ferramenta para reduzir pontos de falhas, como respostas incorretas ou sem fundamento teórico. Novas metodologias, como '*retrieval augmentation*', que incorporam fontes externas para melhorar a precisão das respostas, são discutidas nesse estudo também. O texto termina propondo um framework estruturado para futuras pesquisas e aplicações, consolidando a engenharia de prompts como um campo importante para o avanço da IA. Esse artigo traz alguns pontos bem relevantes para a criação dos testes com contexto deste atual estudo, que ajudam na avaliação das respostas e influenciam no resultado geral.

3.3 Benefícios Potenciais de IA no Ensino

Este estudo [Baidoo-Anu e Ansah \(2023\)](#) explora a literatura recente sobre o uso do ChatGPT no ensino e na aprendizagem, destacando benefícios e desvantagens potenciais. Também aponta limitações como a possibilidade de gerar informações incorretas e preconceitos nos dados de treinamento, sugerindo maneiras de maximizar seu uso educacional. Foi utilizado um modo exploratório para investigar o potencial educacional do ChatGPT. Nele foi mostrado os seguintes pontos: o nível de interatividade do ChatGPT; Os benefícios da GenAI, como o ChatGPT, no avanço da educação; As possíveis desvantagens do uso dessas tecnologias na educação, e considerações e implicações na prática. Os resultados desse artigo trazem que o ChatGPT e outras IA podem revolucionar o sistema educacional. Apesar das preocupações com a forma das respostas geradas, como a possível resolução de informações erradas e preconceitos nos dados, há um desejo crescente por sua aplicação nas escolas. Educadores podem pensar em maneiras de usufruir dessas ferramentas para ajudar os alunos a aprenderem mais facilmente, preparando-os para um futuro cada vez mais influenciado pela IA. No entanto, é importante a discussão entre educadores e especialistas em tecnologia para garantir que seu uso seja seguro e benéfico. O vínculo educacional une este estudo à atual monografia. Ambos têm uma preocupação e um olhar sobre os benefícios e os avanços que o uso de GenAI pode trazer para os alunos, professores e todos os envolvidos no âmbito da educação.

4 Metodologia

Este trabalho foi conduzido com base em estudos, testes de uso e análises exploratórias das ferramentas de GenAI, com foco em possíveis contribuições para o ensino da computação. Com isso, buscou-se avaliar como essas ferramentas poderiam ser utilizadas para facilitar e aumentar o engajamento em atividades de ensino para estudantes e professores. As etapas do estudo incluem:

1. Levantamento dos principais desafios do ensino de programação.
2. Estudo exploratório de 3 ferramentas populares de GenAI.
3. Estudo e análise comparativa das ferramentas.
4. Primeira rodada de testes de uso das ferramentas, envolvendo consultas sobre conceitos essenciais de computação.
5. Segunda rodada de testes com consultas envolvendo exercícios mais voltados para programação.
6. Simulações com entrada de textos em formato PDF e solicitação de ações a partir deles.
7. Análise dos resultados.
8. Propostas de ações, baseados nos desfechos dos testes.

Para atender a todas as etapas, esse capítulo foi separado em cinco seções. A seção 4.1 apresenta o conteúdo dos processos, como foi a preparação e as etapas iniciais do estudo. A seção 4.2 trará a definição e destrinchamento dos critérios de avaliação dos testes. Na seção 4.3 abrange a parte conceitual dos testes da primeira rodada que focam em conceitos fundamentais da computação, em seguida a seção 4.4 com o conceito dos testes da segunda rodada, que se concentram em perguntas mais voltadas especificamente em problemas de programação. Já a seção 4.5 aborda a estruturação dos testes funcionais com alimentação de dados, com a inserção de PDFs e solicitações de resumos e análises desses documentos.

4.1 Processos Iniciais

Inicialmente, para uma melhor aplicação, os primeiros passos foram analisar e comparar algumas ferramentas de IA, com o objetivo de extrair os melhores pontos de cada uma delas. Foi feita uma comparação descritiva para usos específicos na área da computação e tecnologia. Essa análise visou explorar maneiras mais eficazes de utilizar

essas ferramentas para que o ensino de programação e áreas afins da computação se tornem um catalisador de desenvolvimento. Dessa forma, os alunos poderiam compreender claramente cada tópico e evoluir em seus processos pessoais e personalizados de estudo.

Cada ferramenta possui características positivas e negativas, sendo alguns pontos mais relevantes para o estudo, enquanto outras não se encaixam tão bem. Neste estudo, foram escolhidas três ferramentas para comparação: ChatGPT, Gemini e Copilot, que demonstraram maior utilidade nas buscas. Essa investigação proporcionou um entendimento do uso dessas ferramentas em contextos educacionais específicos, explorando suas possibilidades e buscando entender como o aprendizado na área de computação poderiam ser beneficiado.

Para realizar uma comparação mais aprofundada das três ferramentas de GenAI, algumas rodadas de testes com as ferramentas foram realizadas. O processo começou com duas rodadas iniciais de avaliação. A primeira rodada focou em tópicos gerais e teóricos da computação, com o objetivo de extrair respostas que fizessem sentido e fossem corretas a partir das perguntas. Com isso, foi possível avaliar e capturar pontos relevantes sobre conceitos básicos de computação. Na segunda rodada, foram feitas perguntas mais direcionadas à análise e desenvolvimento de códigos e programação. O enfoque principal foi obter uma análise detalhada sobre a capacidade das ferramentas de interpretar e gerar códigos de programação. Essa etapa foi importante para entender como cada ferramenta lida com problemas de codificação simples e mais elaborados, para avaliar suas habilidades.

Para a avaliação, uma lista de perguntas relacionadas à computação foi criada. As perguntas foram elaboradas pensando em conceitos que vão dos mais básicos aos mais avançados, envolvendo disciplinas típicas de um curso de Computação, com base na própria grade do BCC-FACOM-UFU. Nas duas rodadas de teste foram utilizados vários temas. A primeira rodada contou com vinte perguntas, enquanto a segunda rodada tinha dez perguntas um pouco mais complexas por conceito. Para a avaliação das respostas, foram definidos critérios específicos e uma classificação em níveis.

Para a primeira e segunda rodada de testes, foram adotados quatro tipos principais de testes, descritos a seguir:

1. **Agrupados sem contexto:** esses testes envolvem o envio de todas as perguntas de uma vez em um único chat, sem a presença de um texto introdutório ou qualquer tipo de contextualização.
2. **Agrupados com contexto:** nesses testes, todas as perguntas são enviadas juntas em um só chat, mas com a contextualização prévia, ou seja, um texto introdutório é fornecido antes das perguntas para criar um contexto.
3. **Individual sem contexto:** as perguntas são submetidas uma a uma, de forma

individual, sem qualquer informação de contextualização enviada previamente ao chat.

4. **Individual com contexto:** as perguntas são enviadas uma a uma, mas agora com uma introdução enviada previamente antes das perguntas, fornecendo um contexto.

Esses diferentes formatos de testes permitem uma análise detalhada de como a contextualização e o método de envio influenciam as respostas das ferramentas de GenAI.

4.2 Critérios

Para ter um melhor entendimento e parâmetro dos resultados entre as três ferramentas (ChatGPT, Gemini e Copilot), foram estabelecidos alguns critérios com o objetivo de ter uma maior visibilidade das características das tecnologias. Esses critérios tiveram uma definição baseada em pontos importantes que são de grande relevância neste estudo. Os critérios foram avaliados usando uma pontuação de 0 a 5 para cada métrica, facilitando o processo de avaliação. Isso permitiu uma análise mais clara e precisa sobre o que as ferramentas retornaram para cada pergunta. Foram estipulados sete critérios, sendo eles: Desvio do assunto, Corretude, Complexidade, Proximidade da Resposta Esperada, Profundidade, Clareza e Relevância.

1. Desvio do Assunto: Avalia se a resposta se manteve no tópico solicitado. A pontuação varia de 0 (resposta completamente fora do critério) a 5 (resposta totalmente focada no critério).
2. Corretude: Mede se a resposta está tecnicamente correta. A pontuação varia de 0 (resposta totalmente incorreta) a 5 (resposta totalmente correta).
3. Complexidade: Analisa se a complexidade da resposta está adequada à pergunta, garantindo que não seja nem muito simples nem muito complexa. A pontuação varia de 0 (resposta muito simples ou muito complexa, sem adequação ao nível da pergunta) a 5 (resposta perfeitamente ajustada à complexidade requerida pela pergunta).
4. Proximidade da Resposta Esperada: Mede o quão próxima a resposta está daquela que é esperada ou desejada. A pontuação varia de 0 (resposta completamente diferente do esperado) a 5 (resposta exatamente conforme o esperado).
5. Profundidade: Avalia o nível de detalhamento e explicação fornecidos na resposta. A pontuação varia de 0 (resposta sem qualquer profundidade ou explicação) a 5 (resposta extremamente detalhada e abrangente).
6. Clareza: Verifica se a resposta é clara e comprehensível. A pontuação varia de 0 (Resposta completamente confusa e difícil de entender) a 5 (resposta perfeitamente clara e comprehensível).

7. Relevância: Mede se todas as partes da resposta são relevantes à pergunta. A pontuação varia de 0 (resposta totalmente irrelevante para a pergunta) a 5 (resposta completamente relevante, sem partes desnecessárias).

4.3 Primeira Rodada: Conceitos Básicos da Computação

O objetivo desta rodada é avaliar os conceitos básicos da computação, analisando as respostas e qualificando-as segundo cada critério estabelecido anteriormente, além de comparar as respostas entre as diferentes ferramentas. As perguntas aqui foram mais objetivas e teóricas, para estabelecer uma base de temas importantes e relevantes da computação.

Nesta rodada, foram selecionados 10 temas, com 2 perguntas para cada um, totalizando 20 perguntas sobre tópicos específicos. Em seguida as respostas foram avaliadas e foi feita uma análise para os principais destaques. Os temas abordados foram: Programação, Estruturas de Dados, Algoritmos, Sistemas Operacionais, Banco de Dados, Redes de Computadores, Engenharia de Software, Inteligência Artificial, Segurança da Informação e Teoria da Computação. Todas as perguntas podem ser consultadas no Apêndice [A.1](#).

As perguntas foram elaboradas a partir da análise da grade curricular do curso de Ciência da Computação, abordando temas relevantes e as principais dúvidas pesquisadas na área. A partir dos principais tópicos identificados, foram criadas perguntas específicas, que foram submetidas a cada uma das três ferramentas (ChatGPT, Gemini e Copilot). Os resultados obtidos foram analisados, gerando informações sobre o desempenho de cada ferramenta.

No desenvolvimento e processamento, os testes foram realizados em várias etapas. Cada remessa de perguntas foi testada nos chats para cada ferramenta de IA, variando entre perguntas com e sem prompts de contextualização, o que permitiu observar as variações nas respostas conforme essas mudanças. Também foram enviadas de forma individual, uma pergunta de cada vez e todas agrupadas.

Relacionado ao contexto utilizado na primeira rodada, a introdução às perguntas foi pensada como uma explicação do ambiente, informando ao chat que as questões abordariam temas fundamentais de computação. Solicitou-se que o chat fornecesse respostas claras, precisas e explicativas, de forma que tanto iniciantes quanto pessoas mais avançadas pudessem entender. Além disso, foram destacados os requisitos que as respostas deveriam atender. Segue o trecho de contextualização do chat:

Essas perguntas contêm conceitos fundamentais da computação e são específicas para chats de IAs para avaliar sua capacidade de fornecer respostas claras, precisas e explicativas, podendo servir tanto para iniciantes quanto para alunos com experiência. Cada pergunta foi criada para explorar não só o critério de profundidade técnica das respostas, mas também a clareza e a acessibilidade das explicações propostas.

as respostas devem atender aos requisitos:

- explicar o conceito básico de cada pergunta com exatidão
- fornecer exemplos claros e, se possível, oferecer comparação para ajudar na compreensão.
- manter uma escrita compreensível e acessível, evitando duplicidade e jargões e detalhando os termos importantes

Esta estratégia de contextualização está relacionada à engenharia de prompt, que busca obter respostas mais completas e precisas por meio de perguntas bem estruturadas e de uma explicação clara do contexto. Dessa forma, é possível alinhar exatamente o que se deseja com o que realmente é necessário obter na resposta, incluindo todas as informações relevantes antecipadamente. Como citado na seção 3.3, a engenharia de prompt pode ser extremamente útil para fins de estudo, e agora, na prática, será aplicado esse conceito em diferentes formas de teste para avaliar seu desempenho.

4.4 Segunda Rodada: Conceitos Específicos de Programação

Esta rodada foca em tópicos relacionados à programação, com perguntas específicas direcionadas a temas que envolvem códigos e linguagens de programação.

Nesta rodada, temos 10 perguntas de programação, abrangendo níveis variados de dificuldade. As perguntas foram projetadas para atender alunos com pouco conhecimento no assunto, bem como aqueles com um entendimento mais profundo. Os tópicos abordados incluem: Imprimindo uma Mensagem, Variáveis e Operações Aritméticas, Estruturas Condicionais, Laços de Repetição, Listas, Funções, Manipulação de Strings, Dicionários, Classes e Objetos, e Algoritmos de Ordenação. Todas as perguntas podem ser consultadas no Apêndice [A.2](#).

As perguntas de programação foram desenvolvidas para cobrir uma ampla gama de exercícios comuns durante o período de aprendizagem. Essas questões foram submetidas a cada ferramenta de IA em comparação. Nesta rodada de testes, utilizamos o mesmo procedimento da rodada anterior. Primeiro, todas as perguntas foram enviadas de uma só vez em um mesmo chat, sem contextualização. Em seguida, foi feito um teste enviando as perguntas uma a uma, ainda sem contextualização. Em uma etapa posterior, foi introduzido um contexto explicativo, indicando os tipos de perguntas e os critérios esperados nas respostas. Esses testes foram realizados com todas as perguntas agrupadas e individualmente. Por fim, aplicamos o teste de configurações personalizadas com alterações específicas de algumas ferramentas que tinha suporte gratuito.

No contexto da segunda rodada, o enfoque foi semelhante ao da primeira, com a alteração de um dos critérios, incluindo a solicitação de exemplos sempre que possível e códigos completos. Foram fornecidos temas fundamentais da programação e solicitado ao chat que as respostas fossem acessíveis, claras e bem explicadas, atendendo a públicos

de diversos níveis. Além disso, foram destacados os requisitos que as respostas deveriam cumprir. Segue o trecho de contextualização do chat:

Essas perguntas contêm conceitos fundamentais da programação e são específicas para chats de IAs para avaliar sua capacidade de fornecer respostas claras, precisas e explicativas, podendo servir tanto para iniciantes quanto para alunos com experiência. Cada pergunta foi criada para explorar não só o critério de profundidade técnica das respostas, mas também a clareza e a acessibilidade das explicações propostas.

as respostas devem atender aos requisitos:

- explicar o conceito básico de cada pergunta com exatidão
- fornecer exemplos claros e, sempre que possível, oferecer o código completo.
- manter uma escrita compreensível e acessível, evitando duplicidade e jargões e detalhando os pontos importantes

Na segunda rodada os testes foram realizados logo em sequência da primeira rodada, os resultados foram coletados e analisados. A pontuação para cada resposta foi aplicada, conforme os tipos de testes e associada aos critérios de avaliação.

4.5 Testes Especiais

Foi realizado um grupo de testes fora dos padrões anteriores, como aqueles que envolveram perguntas associadas a uma configuração específica da ferramenta. Apenas o ChatGPT oferecia essa personalização de forma gratuita, o que representou um diferencial em relação às outras ferramentas. Outro tipo de teste envolveu conteúdos de PDFs, para verificar como as ferramentas processavam e forneciam informações a partir desses arquivos. Foi possível observar que, no momento deste estudo, nem todas as ferramentas tinham o recurso de submissão de PDFs. Apenas o ChatGPT suportava essa funcionalidade, enquanto o Gemini e Copilot não permitia, apenas a análise de imagens e textos, sem suporte a PDFs.

Esse teste seguiu exatamente esse princípio. Foi adicionado uma breve descrição de perfil, como estudante de ciência da computação de 26 anos, e o setor de atuação do mercado de trabalho. Além disso, foi acrescentado um campo com especificações para as respostas do chat, utilizando um texto semelhante ao dos testes com contextualização, solicitando respostas claras, precisas e explicativas, com exemplos sempre que possível. Essas descrições foram configuradas em inglês, de acordo com as configurações de idioma do chat. a figura 1 demonstra as definições.

Novos artigos, livros e conteúdos estão constantemente surgindo, mas atualmente, as ferramentas de IA ainda não conseguem ser alimentadas em tempo real com essas informações. Foi realizado um grupo de testes focado no conhecimento de informações recentes, ou seja, elementos desconhecidos pelas IAs, como novos artigos. A introdução de

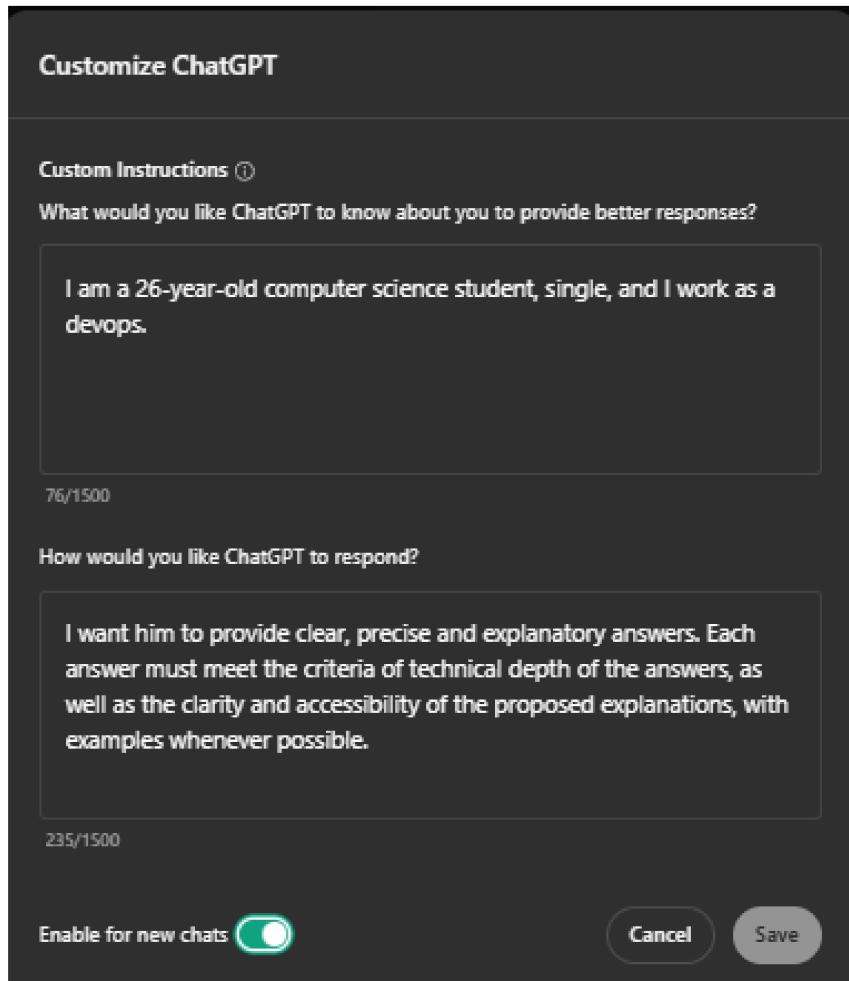


Figura 1 – Configuração Personalizada do ChatGPT

PDFs e perguntas baseadas nesses documentos, bem como a geração automática de questões a partir do conteúdo, abre diversas possibilidades no campo educacional, oferecendo valor para o ensino de programação.

Os testes com alimentação de dados de PDFs foram realizados em três diferentes modalidades de abordagem: Respostas Relacionadas a Dúvidas Gerais, em que foi submetido um PDF e foram feitas perguntas sobre seu conteúdo; Geração de conteúdo a Partir de PDF, em que foram geradas perguntas baseadas no conteúdo do PDF; e Validação de Questões sobre Programação, onde um PDF de referência foi comparado com outro PDF para verificar a precisão e coerência das respostas.

5 Resultados e Análises

Após a execução dos testes com as ferramentas de IA, foram obtidos resultados e impressões importantes que permitem orientar as escolhas para um melhor uso das ferramentas, alinhando com as necessidades e objetivos específicos dos interessados.

A comparação entre o ChatGPT, Gemini e Copilot buscou destacar suas limitações e possibilidades de uso no ensino de computação, considerando os contextos testados.

Para cada ferramenta de GenAI, nas duas rodadas, foi criado um gráfico para análise geral, utilizando como parâmetros os principais tipos de testes e os sete critérios de avaliação utilizados nos testes. Os gráficos foram plotados a partir da pontuação de cada teste, calculando a média dos valores pontuados e agrupados por tipo de teste e critério. Esse processo foi realizado para cada ferramenta em ambas as rodadas e os testes especiais.

5.1 Resultados da Primeira Rodada de Testes

Nessa fase, foram feitas várias perguntas relacionadas aos principais temas da computação, tópicos que os alunos geralmente encontram ao longo do curso. As respostas de cada IA foram avaliadas de acordo com critérios mencionados no item 4.2, proporcionando uma compreensão mais clara das características de cada ferramenta.

As três ferramentas tiveram resultados bem relevantes na primeira rodada de perguntas sobre tópicos mais gerais. Em geral, as ferramentas não fugiram ou desviaram tanto do assunto, tendo obtido boas notas neste quesito. As respostas geralmente eram corretas, sem erros conceituais, embora em alguns momentos tenham sido notadas omissões na explicação dos conceitos.

O ChatGPT foi o que obteve as pontuações mais altas, com respostas completas e assertivas, às vezes com excesso de informação, mas frequentemente com alta relevância. O Gemini e o Copilot tiveram resultados bem semelhantes, porém o Gemini se destacou mais, pois suas respostas apresentavam um pouco mais de profundidade em comparação ao Copilot.

Outro ponto a ser considerado é como as perguntas foram feitas. Nota-se que foi criado um chat em cada ferramenta, onde todas as perguntas foram inseridas de uma vez, e as respostas foram aguardadas. Também foram inseridas individualmente. Em alguns testes, o Gemini apresentou um nível maior de detalhamento, quando as perguntas foram enviadas separadamente e com uma quantidade menor de caracteres. O ChatGPT trouxe ainda mais pontos, que muitas vezes não eram tão relevantes, quando as perguntas foram

enviadas de forma independente. O Copilot ofereceu mais detalhes em perguntas enviadas separadamente.

5.1.1 ChatGPT

A figura 2 revela que os resultados do ChatGPT foram, em sua maioria, relevantes, sem grandes trechos considerados irrelevantes, como a métrica o desvio no assunto pode mostrar. No critério de complexidade e profundidade, os testes realizados de forma agrupada e sem contextualização apresentaram um desempenho significativamente inferior em comparação aos outros formatos. Nota-se que os testes individuais mostraram uma vantagem clara sobre os testes agrupados, com pontuações mais elevadas. Um ponto interessante é que, nos testes agrupados, as respostas sem contextualização inicial tiveram uma performance melhor em comparação às respostas com contexto. Já nos testes individuais, tanto com quanto sem contextualização, as variações foram mínimas.

Gráfico critérios por tipo de teste ChatGPT

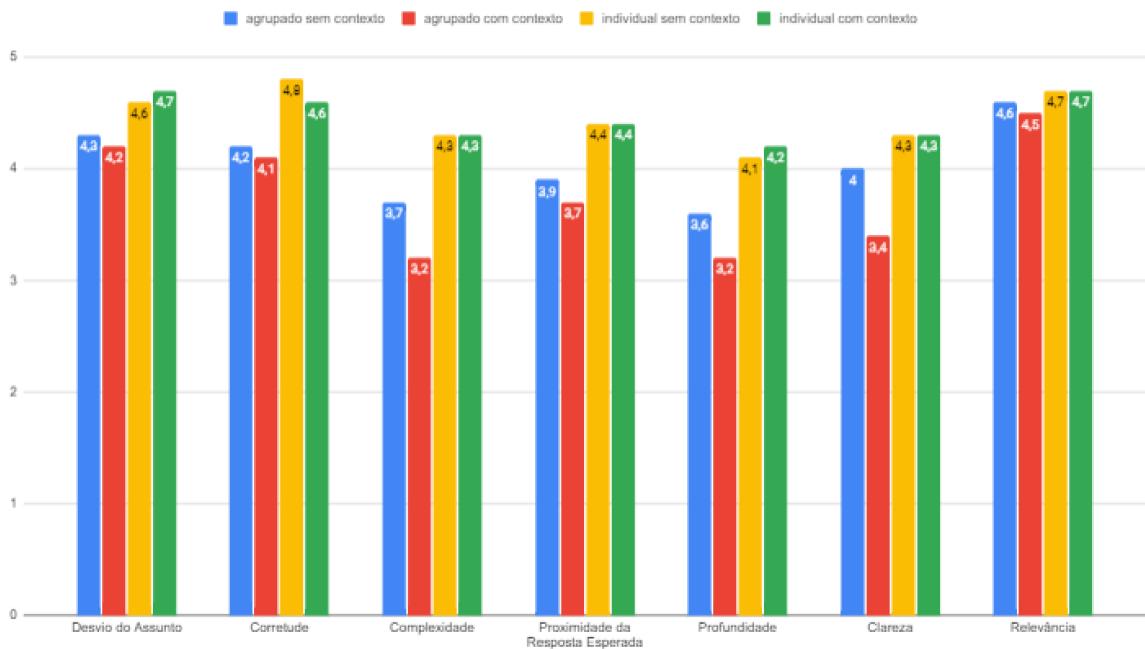


Figura 2 – Figura do ChatGPT Critérios por Tipos de Testes

Além disso, é importante destacar algumas respostas que chamaram a atenção positivamente e agregaram valor à classificação do ChatGPT, revelando características e padrões comuns em seu desempenho. Três respostas específicas merecem ser ressaltadas.

Primeiro exemplo: Perguntas relacionadas à IA.

Houve uma notável discrepância entre a resposta do ChatGPT para a questão de IA sobre aprendizado supervisionado e não supervisionado, no teste de perguntas agrupadas com contextualização e no teste de perguntas individuais sem contextualização.

A figura 3 ilustra a resposta obtida no primeiro caso, enquanto a figura 4 mostra a resposta do segundo cenário. No teste de perguntas agrupadas, a resposta foi sucinta e um tanto superficial, com alguns pontos não totalmente claros, como a definição do algoritmo de busca A* com uma resolução mais genérica e sem muita profundidade. Em contraste, a resposta das perguntas enviadas individualmente trouxe um salto significativo na qualidade: a definição sobre os tipos de aprendizado foi detalhada, incluindo exemplos claros e a estruturação em tópicos que elevavam a definição, objetivo, exemplos práticos e algoritmos comuns. Já a resposta sobre o algoritmo de busca A* apresentou, além de uma descrição minuciosa, explicações sobre o funcionamento, o cálculo de custo, vantagens, e exemplos práticos que ilustravam sua aplicação.

Inteligência Artificial

15. O que é aprendizado supervisionado e não supervisionado em machine learning?

- No aprendizado supervisionado, um modelo é treinado com um conjunto de dados rotulados, onde a entrada e a saída desejada são conhecidas. O objetivo é que o modelo aprenda a prever a saída a partir da entrada.
- No aprendizado não supervisionado, o modelo trabalha com dados não rotulados e busca padrões ou agrupamentos dentro dos dados. Um exemplo é a segmentação de clientes com base em comportamentos.

16. Explique o que é um algoritmo de busca A*

- O algoritmo de busca A* é um algoritmo de busca heurística utilizado para encontrar o caminho mais curto em grafos. Ele combina as abordagens de busca em largura e busca de custo uniforme, utilizando uma função de avaliação que considera o custo do caminho percorrido e uma estimativa do custo restante até o destino.

Figura 3 – Resposta de pergunta agrupada no ChatGPT sobre IA

Segundo exemplo: Perguntas relacionadas à algoritmos.

Neste foi comparado dois testes com perguntas individuais, mas agora diferenciando entre respostas com e sem contexto. No teste sem contexto (figura 5), o desempenho foi satisfatório, apresentando respostas que atendiam ao que foi solicitado, embora sem grande profundidade. A resposta destacou-se pela objetividade. O ChatGPT trouxe uma definição sobre a complexidade de tempo, explicando sua relevância em parágrafos bem estruturados. Além disso, fez uma comparação entre a busca linear e a busca binária, oferecendo exemplos práticos que ilustravam as diferenças de maneira acessível. Por outro lado, no teste com contexto (figura 6), o ChatGPT apresentou uma abordagem diferente: uma série de pontos que descreviam detalhadamente cada notação de complexidade, acompanhados de trechos de código em Python para exemplificar as definições. Além disso, a comparação foi enriquecida com explicações e um exemplo final que ajudava no entendimento sobre o assunto, bem mais extenso que o sem contexto.

 15. O que é aprendizado supervisionado e não supervisionado em machine learning?

Aprendizado Supervisionado e Aprendizado Não Supervisionado são duas abordagens principais em machine learning (aprendizado de máquina), que diferem na maneira como os dados são usados para treinar os modelos.

- Aprendizado Supervisionado:
 - Definição: No aprendizado supervisionado, o modelo é treinado usando um conjunto de dados rotulado, ou seja, um conjunto de entrada onde as respostas corretas (rótulos) já são conhecidas.
 - Objetivo: O modelo aprende a mapear as entradas para as saídas corretas, com base nos exemplos fornecidos. A principal tarefa é fazer previsões ou classificações com base nos dados de entrada.
 - Exemplo: Classificação de e-mails como "spam" ou "não spam" (rótulos conhecidos), ou prever o preço de uma casa com base em variáveis como tamanho, localização, etc.
 - Algoritmos comuns: Regressão linear, máquinas de vetores de suporte (SVM), redes neurais, árvores de decisão, etc.
- Aprendizado Não Supervisionado:
 - Definição: No aprendizado não supervisionado, o modelo é treinado com dados que não têm rótulos. O objetivo é identificar padrões ou estruturas ocultas nos dados, sem a necessidade de rótulos para as saídas.
 - Objetivo: O modelo tenta entender a estrutura subjacente dos dados por si mesmo. Isso pode incluir agrupamento, redução de dimensionalidade ou detecção de anomalias.
 - Exemplo: Segmentação de clientes com base em comportamentos de compra semelhantes (agrupamento de dados), ou redução de dimensões para visualizar dados em espaços 2D ou 3D.
 - Algoritmos comuns: K-means, algoritmos de agrupamento hierárquico, análise de componentes principais (PCA), mapas auto-organizáveis (SOM), etc.

Diferença principal:

- O aprendizado supervisionado requer rótulos ou respostas conhecidas para treinar o modelo, enquanto o aprendizado não supervisionado trabalha com dados sem rótulos e tenta descobrir padrões ou estruturas nos dados.

 16. Explique o que é um algoritmo de busca A*.

O algoritmo A* é um algoritmo de busca utilizado para encontrar o caminho mais curto (ou o mais eficiente) entre dois pontos em um grafo, com base em uma combinação de custo do caminho percorrido e uma estimativa heurística do custo restante até o objetivo. Ele é amplamente utilizado em problemas de navegação, como mapas de rotas, jogos e robótica.

- Funcionamento: O A* combina as vantagens de dois outros algoritmos de busca: Dijkstra e Busca Greedy:
 - Dijkstra garante a solução ótima, mas explora de forma cega todas as possibilidades de caminhos.
 - Busca Greedy: tenta encontrar a solução rapidamente, mas sem garantir que será a melhor.
- O A* faz isso utilizando uma função de custo total, que é a soma de duas partes:
 - $g(n)$: o custo do caminho já percorrido até o nó atual n (conhecido como custo acumulado).
 - $h(n)$: uma função heurística que estima o custo restante do nó atual até o objetivo. Essa função ajuda o algoritmo a direcionar sua busca na direção mais promissora.

A fórmula do A* para calcular o custo de cada nó é: $f(n) = g(n) + h(n)$

Onde:

- $f(n)$ é o custo total estimado do nó n .
- $g(n)$ é o custo do caminho até o nó n , a partir do nó inicial.
- $h(n)$ é a estimativa heurística do custo restante do nó n até o objetivo.

• Passos do algoritmo:

1. O algoritmo começa no nó inicial e explora os vizinhos.
2. Para cada nó, calcula o valor de $f(n)$ e prioriza a expansão do nó com o menor valor de $f(n)$.
3. O A* continua explorando os nós até alcançar o nó objetivo ou até que não haja mais nós para explorar.

• Heurística: A escolha de uma boa heurística é crucial para a eficiência do A*. A heurística deve ser admissível, ou seja, nunca deve superestimar o custo real para alcançar o objetivo, para garantir que o A* encontre uma solução ótima.

Vantagens do A*:

- Garante encontrar o caminho mais curto se a heurística for admissível.
- É mais eficiente do que a busca cega (como Dijkstra) devido à heurística, que direciona a busca.

Exemplo: Em um jogo de labirinto, o A* pode ser usado para encontrar o caminho mais rápido do ponto inicial até a saída, considerando tanto a distância percorrida até agora quanto uma estimativa da distância restante até o objetivo.

Figura 4 – Resposta de pergunta individual no ChatGPT sobre IA

5.1.2 Gemini

A figura 7 ilustra detalhadamente o desempenho do Gemini em diversos critérios de avaliação das respostas. De forma geral, os testes feitos de maneira individual com contexto apresentaram as melhores pontuações em quase todos os pontos analisados, destacando-se especialmente em desvio no assunto, corretude, complexidade, profundidade, clareza e relevância. Esses resultados indicam que a introdução de um enunciado prévio ajuda o Gemini a oferecer respostas mais completas e precisas.

Os testes individuais sem contexto também se destacaram em alguns critérios, como desvio de assunto, corretude e proximidade com a resposta esperada. Essa performance sugere que o formato individual, mesmo sem contexto, permite que o Gemini tenha um bom entendimento das questões e produza respostas adequadas, embora possam faltar exemplos mais elaborados e uma análise mais aprofundada. Em contrapartida, os testes agrupados, tanto com quanto sem contexto, tiveram um desempenho inferior, especialmente nos critérios de complexidade, profundidade e clareza. Isso sugere que a falta de contexto e o envio em massa de perguntas impactam a eficácia e o detalhamento das respostas do Gemini. Contudo, nos critérios de relevância, os testes agrupados mostraram uma vantagem: as respostas eram mais objetivas e focadas apenas no solicitado,

5. O que é a complexidade de tempo de um algoritmo e por que é importante?

A complexidade de tempo de um algoritmo é uma medida que indica a quantidade de tempo que ele leva para ser executado em função do tamanho da entrada (geralmente denotada por n). Ela é expressa usando a notação Big-O, como $O(n)$, $O(\log n)$, $O(n^2)$, etc., que descreve o comportamento do tempo de execução à medida que o tamanho dos dados aumenta.

A complexidade de tempo é importante porque ajuda a avaliar a eficiência de um algoritmo. Em problemas com grandes conjuntos de dados, um algoritmo com complexidade de tempo menor pode resolver o problema muito mais rapidamente do que um algoritmo com complexidade maior. Assim, ao entender a complexidade, é possível escolher um algoritmo mais apropriado para o contexto e garantir que o programa seja escalável e execute de forma eficiente.

6. Explique a diferença entre busca linear e busca binária.

A busca linear e a busca binária são duas técnicas de pesquisa em listas de elementos, mas diferem na forma de operar e nos tipos de listas em que são eficazes.

- Busca Linear:** nesta abordagem, o algoritmo percorre a lista sequencialmente, elemento por elemento, até encontrar o valor desejado ou até o final da lista. Ela funciona em qualquer tipo de lista, ordenada ou não, mas é menos eficiente para listas grandes, pois sua complexidade de tempo é $O(n)$.
 - Exemplo: em uma lista de 10 elementos, uma busca linear pode levar até 10 comparações no pior caso.
- Busca Binária:** neste caso, o algoritmo só funciona em listas ordenadas. Ele começa comparando o valor do meio da lista com o valor desejado. Se o valor procurado for menor, ele descarta a metade superior; se for maior, descarta a metade inferior. Esse processo continua até encontrar o elemento ou concluir que ele não está presente. A busca binária é mais eficiente do que a linear para listas grandes, com complexidade de tempo $O(\log n)$.
 - Exemplo: em uma lista ordenada de 1.000 elementos, uma busca binária só precisa de, no máximo, cerca de 10 comparações no pior caso.

Resumo:

- Busca Linear:** verifica cada elemento, usada para listas desordenadas, $O(n)$.
- Busca Binária:** descarta metade da lista a cada passo, usada para listas ordenadas, $O(\log n)$.

Figura 5 – Resposta de pergunta sem contexto no ChatGPT sobre algoritmo

Aqui estão as respostas para suas perguntas sobre algoritmos:

5. O que é a complexidade de tempo de um algoritmo e por que é importante?

A complexidade de tempo de um algoritmo é uma medida de quão rapidamente o tempo de execução de um algoritmo aumenta à medida que o tamanho da entrada cresce. Em outras palavras, ela descreve o comportamento do tempo de execução do algoritmo com base no tamanho dos dados que ele processa. A complexidade de tempo é geralmente expressa usando a notação Big-O, que descreve o pior caso, ou seja, a quantidade máxima de tempo que o algoritmo pode levar para processar os dados.

Exemplo de notações de complexidade de tempo:

- $O(1)$: Tempo constante. O algoritmo leva o mesmo tempo para executar, independentemente do tamanho da entrada. Exemplo: acessar um elemento em um array.
- $O(n)$: Tempo linear. O tempo de execução cresce proporcionalmente ao tamanho da entrada. Exemplo: percorrer todos os elementos de uma lista.
- $O(n^2)$: Tempo quadrático. O tempo de execução cresce proporcionalmente ao quadrado do tamanho da entrada. Exemplo: algoritmos de ordenação simples como a ordenação por bolha.
- $O(\log n)$: Tempo logarítmico. O tempo de execução cresce de forma logarítmica em relação ao tamanho da entrada. Exemplo: busca binária.

Importância da complexidade de tempo:

- Eficiência:** A complexidade de tempo ajuda a entender a eficiência de um algoritmo. Algoritmos mais rápidos são preferíveis, especialmente quando lidamos com grandes volumes de dados.
- Escalabilidade:** Conhecer a complexidade de tempo é importante para garantir que um algoritmo continue a funcionar de maneira eficiente mesmo com o aumento do tamanho dos dados.
- Otimização:** A análise da complexidade de tempo permite identificar e melhorar pontos de gargalo em um programa, tornando-o mais rápido e eficiente.

6. Explique a diferença entre busca linear e busca binária.

Busca Linear: A busca linear é um algoritmo simples que percorre todos os elementos de uma lista ou array, verificando um a um se encontra o valor procurado. Ela é eficaz quando a lista está desordenada ou quando não se pode fazer suposições sobre a ordem dos dados.

- Funcionamento:** O algoritmo começa no início da lista e verifica cada elemento sequencialmente até encontrar o valor ou atingir o final da lista.
- Complexidade:** A complexidade de tempo da busca linear é $O(n)$, onde n é o número de elementos na lista. No pior caso, todos os elementos podem ser verificados.

Exemplo de busca linear:

```
python
# Exemplo de busca linear
def busca_linear(lista, valor):
    for i in range(len(lista)):
        if lista[i] == valor:
            return i # Retorna o índice do valor encontrado
    return -1 # Retorna -1 se o valor não foi encontrado
```

Busca Binária: A busca binária é um algoritmo eficiente para procurar um valor em uma lista ordenada. Em cada passo, ela divide a lista pela metade, comparando o valor procurado com o elemento no meio da lista. Com isso, elimina metade dos elementos, reduzindo significativamente o número de comparações.

- Funcionamento:** A busca binária começa com a lista ordenada e verifica o elemento do meio. Se o valor procurado for menor que o elemento do meio, a busca continua na metade inferior da lista; caso contrário, continua na metade superior. Esse processo se repete até o valor ser encontrado ou a lista ser reduzida a um único elemento.
- Complexidade:** A complexidade de tempo da busca binária é $O(\log n)$, onde n é o número de elementos na lista. A busca binária é muito mais eficiente que a busca linear, especialmente para listas grandes, já que ela reduz o número de comparações de forma exponencial.

Exemplo de busca binária:

```
python
# Exemplo de busca binária
def busca_binaria(lista, valor):
    inicio, fim = 0, len(lista) - 1
    while inicio <= fim:
        meio = (inicio + fim) // 2
        if lista[meio] == valor:
            return meio # Retorna o índice do valor encontrado
        elif lista[meio] < valor:
            inicio = meio + 1 # Busca na metade superior
        else:
            fim = meio - 1 # Busca na metade inferior
    return -1 # Retorna -1 se o valor não foi encontrado
```

Comparação entre busca linear e busca binária:

- Ordenação:** A busca linear funciona em listas desordenadas, enquanto a busca binária só funciona em listas ordenadas.
- Eficiência:** A busca binária é muito mais eficiente em listas grandes, com uma complexidade de $O(\log n)$, enquanto a busca linear tem complexidade $O(n)$.
- Uso:** A busca linear é simples e pode ser usada em qualquer tipo de lista, enquanto a busca binária exige uma lista ordenada e é mais eficiente quando o número de elementos é grande.

Esses conceitos ajudam a entender como diferentes algoritmos de busca funcionam e como escolher o mais adequado dependendo do contexto. Se precisar de mais explicações ou exemplos, estou à disposição!

Figura 6 – Resposta de pergunta com contexto ChatGPT sobre algoritmo

sem informações adicionais desnecessárias.

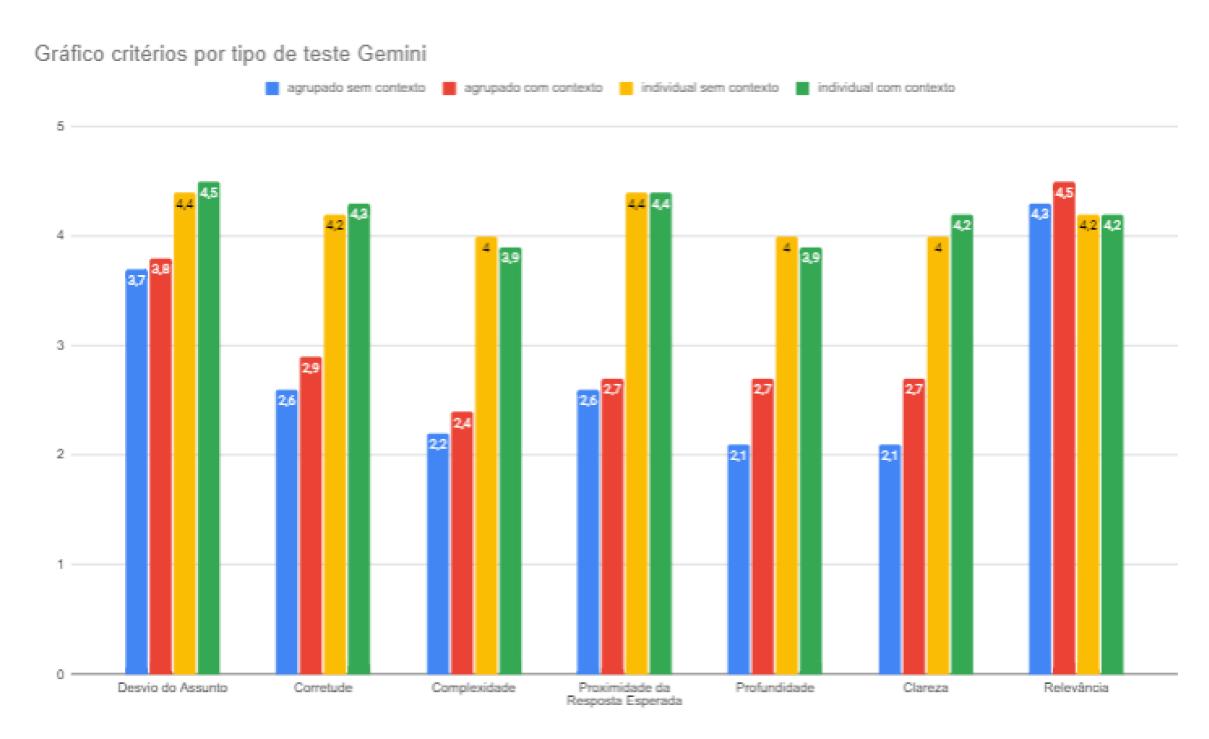


Figura 7 – Figura do Gemini de critérios por tipo de teste

As resoluções no Gemini apresentaram grandes diferenciais, com um conteúdo diversificado e bem estruturado. De modo geral, ele forneceu respostas bastante completas, especialmente quando as perguntas foram feitas de forma individual e com contextualização prévia. Além de explicações detalhadas e exemplos, o Gemini se destacou por incluir informações em diferentes formatos, como tabelas comparativas e tabelas de exemplos, que enriqueceram a compreensão do conteúdo.

Outro ponto forte foi a adição de elementos multimídia, como imagens e links para sites e vídeos no YouTube, fornecendo recursos adicionais para aprofundar o entendimento das respostas. Essa abordagem faz com que o Gemini se destaque como uma boa opção para quem busca não apenas respostas diretas, mas também uma diversidade de conteúdo que pode complementar o aprendizado e trazer uma perspectiva mais ampla sobre o tema.

Nas imagens 8 e 9, podemos observar a diferença nas respostas do Gemini para o tópico de estrutura de dados em diferentes contextos. Na resposta agrupada, ele apresentou uma solução mais curta e direta, abordando as diferenças entre array, lista, pilha e fila de forma resumida, condensada em dois blocos de parágrafos. Por outro lado, na resposta individual sem contextualização, o Gemini ofereceu uma análise muito mais detalhada, desctrinchando cada comparação em vários tópicos, como estrutura, acesso, tamanho, inserção e remoção. Além disso, ele incluiu exemplos práticos para comparar arrays e listas ligadas. Na parte que comparou filas e pilhas, a resposta seguiu uma estrutura similar,

apresentando detalhes e exemplos esclarecedores. O diferencial foi a inclusão de uma imagem e um link para um site com conteúdo aprofundado e ilustrações, que enriqueceu ainda mais a resposta. Essa abordagem mais detalhada e visualmente suportada mostra que, em contextos individuais, o Gemini pode oferecer um nível de detalhamento e recursos adicionais que são bastante valiosos para o aprendizado.

Estruturas de Dados

3. Array vs. Lista Ligada: Um **array** é uma coleção de elementos de mesmo tipo, armazenados em posições consecutivas na memória. Uma **lista ligada** é uma estrutura de dados dinâmica onde cada elemento (nó) contém um valor e um ponteiro para o próximo nó. Arrays são mais eficientes para acesso aleatório, enquanto listas ligadas são mais flexíveis para inserção e remoção de elementos.

4. Pilha e Fila: Uma **pilha** segue o princípio LIFO (Last In, First Out), ou seja, o último elemento inserido é o primeiro a ser removido. Imagine uma pilha de pratos: o último prato colocado é o primeiro a ser retirado. Uma **fila** segue o princípio FIFO (First In, First Out), como uma fila de espera: o primeiro elemento inserido é o primeiro a ser removido.

Figura 8 – Resposta de pergunta agrupada do Gemini sobre estrutura de dados

Estruturas de Dados: Um mergulho mais profundo

3. Array vs. Lista Ligada

Array (Vetor):

- Estrutura: Sequência de elementos do mesmo tipo, armazenados em posições de memória contíguas.
- Acesso: Acesso direto a qualquer elemento pelo índice (posição).
- Tamanho: Fixo, definido na criação.
- Inserção/Remoção: Pode ser custosa, especialmente no meio do array, pois envolve deslocar elementos.
- Exemplo: Um array de inteiros para armazenar as notas de uma turma.

Lista Ligada:

- Estrutura: Coleção de elementos (nós) onde cada nó contém um dado e um ponteiro para o próximo nó.
- Acesso: Acesso sequencial, percorrendo a lista do início ao fim.
- Tamanho: Dinâmico, pode crescer ou diminuir conforme necessário.
- Inserção/Remoção: Geralmente mais eficiente, especialmente no meio da lista, pois envolve apenas a atualização de alguns ponteiros.
- Exemplo: Uma lista de contatos, onde cada contato é um nó da lista.

Diferenças chave:

Característica	Array	Lista Ligada
Estrutura de memória	Contígua	Dispersa
Acesso	Dirto por índice	Sequencial
Tamanho	Fixo	Dinâmico
Inserção/remoção	Pode ser custosa	Geralmente eficiente

4. Pilha (Stack) e Fila (Queue)

Pilha:

- Estrutura: Segue o princípio LIFO (Last In, First Out), ou seja, o último elemento adicionado é o primeiro a ser removido.
- Operações:
 - Push: Adiciona um elemento ao topo da pilha.
 - Pop: Remove o elemento do topo da pilha.
 - Peek: Retorna o elemento do topo sem removê-lo.
- Exemplo: Pilha de pratos, pilha de chamadas de funções.

Fila:

- Estrutura: Segue o princípio FIFO (First In, First Out), ou seja, o primeiro elemento adicionado é o primeiro a ser removido.
- Operações:
 - Enqueue: Adiciona um elemento ao final da fila.
 - Dequeue: Remove o elemento do inicio da fila.
 - Peek: Retorna o elemento do inicio da fila sem removê-lo.
- Exemplo: Fila de espera em um banco, fila de impressão.

Visualização:

Em resumo:

Arrays, listas ligadas, pilhas e filas são estruturas de dados fundamentais em programação. A escolha da estrutura adequada depende dos requisitos do problema a ser resolvido. Ao compreender as características e aplicações de cada uma, você poderá escrever algoritmos mais eficientes e elegantes.

Gostaria de explorar mais sobre algumas dessas estruturas de dados ou aprender sobre outras, como árvores e grafos?

Implementação:

Ambas as estruturas podem ser implementadas usando arrays ou listas ligadas. A escolha da implementação depende dos requisitos específicos da aplicação.

Quando usar cada uma?

- **Arrays:** Quando você precisa de acesso rápido a elementos por índice e o tamanho da estrutura é conhecido antecipadamente.
- **Listas Ligadas:** Quando você precisa inserir e remover elementos frequentemente no meio da estrutura e o tamanho não é conhecido.
- **Pilhas:** Em situações onde a ordem de acesso aos elementos é importante, como em backtracking, avaliação de expressões aritméticas, gerenciamento de memória.
- **Filas:** Em situações onde a ordem de chegada dos elementos é importante, como em sistemas de filas de espera, buffers de comunicação.

Figura 9 – Resposta de pergunta individual do Gemini sobre estrutura de dados

5.1.3 Copilot

Na figura dos critérios de testes do Copilot (figura 10), observa-se que o tipo de teste 'individual com contexto' tem desempenho notavelmente superior em quase todos os critérios. Ele apresenta as maiores notas para desvio do assunto, corretude, complexidade, proximidade da resposta esperada, profundidade e clareza, o que sugere que testes individuais incrementados com contexto produzem resultados mais cheios e completos. As

categorias 'agrupado sem contexto' e 'individual sem contexto' também mostram bons desempenhos em alguns critérios, como relevância e clareza, mas ficam atrás do modelo 'individual com contexto'. O desempenho mais baixo foi observado em 'agrupado com contexto', particularmente em critérios como complexidade, clareza e profundidade, indicando que a ausência de contexto prejudica a eficácia dos testes, principalmente em grupo. A relevância é consistente em todos os modelos, com variações mínimas, mostrando que é um critério bem avaliado em geral.

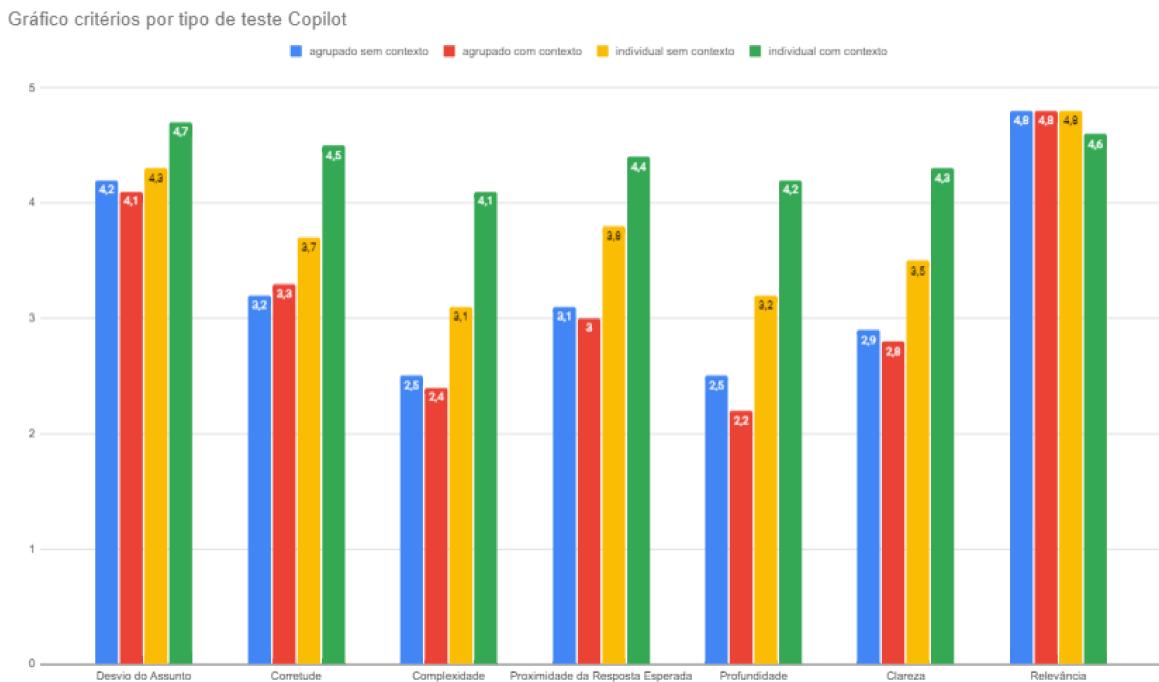


Figura 10 – Figura do Copilot de critérios por tipo de teste

Na figura, pode-se observar que o Copilot teve um desempenho satisfatório em respostas com contextualização. Comparando as respostas individuais com e sem contexto, há diferenças notáveis, como no caso da pergunta sobre estruturas de dados. Na resposta sem contexto (figura 11), o Copilot apresentou uma explicação mais conceitual, abordando os pontos principais e entregando uma resposta de qualidade. Entretanto, na resposta com contexto (figura 12), ele foi além e ofereceu uma ampla gama de detalhes. Trouxe exemplos de código completos e funcionais, enriquecendo o entendimento do conceito e proporcionando uma visão mais prática e aplicada do tema. Isso evidencia que, ao receber um contexto detalhado, o Copilot pode entregar respostas mais abrangentes e informativas, tornando-se uma ferramenta ainda mais eficaz para quem busca aprofundamento em tópicos complexos de programação.

Outro exemplo interessante no Copilot é a análise das respostas sobre IA, comparando sem contexto (figura 13) e com contexto (figura 14), ambas na categoria de

Arrays e Listas Ligadas

Um array é uma estrutura de dados que armazena elementos de tamanho fixo, permitindo acesso direto a qualquer posição do array, pois todos os elementos estão contíguos na memória. Isso torna a leitura e a atribuição de valores extremamente rápidas. No entanto, a inserção e a remoção de elementos podem ser lentas porque muitas vezes exigem deslocar vários elementos.

Uma lista ligada é uma coleção de elementos onde cada elemento (nó) contém um valor e uma referência (ponteiro) ao próximo nó na sequência. Existem vários tipos de listas ligadas, como listas simplesmente ligadas, listas duplamente ligadas e listas circularmente ligadas. Diferentemente dos arrays, as listas ligadas não armazenam elementos contíguos na memória, o que permite inserções e remoções eficientes em qualquer posição da lista. No entanto, o acesso aos elementos pode ser mais lento, pois é necessário percorrer a lista a partir do primeiro nó.

Pilha (Stack) e Fila (Queue)

Uma pilha (stack) é uma estrutura de dados que segue o princípio LIFO (Last In, First Out), ou seja, o último elemento a ser inserido é o primeiro a ser removido. Pensa-se em uma pilha de pratos: você sempre tira o prato de cima. As operações básicas são:

- push: adiciona um elemento ao topo da pilha
- pop: remove o elemento do topo da pilha
- peek: consulta o elemento no topo sem removê-lo

Uma fila (queue), por outro lado, segue o princípio FIFO (First In, First Out), ou seja, o primeiro elemento a ser inserido é o primeiro a ser removido. Pensa-se em uma fila de pessoas esperando: a primeira pessoa a entrar na fila é a primeira a sair. As operações básicas são:

- enqueue: adiciona um elemento ao fim da fila
- dequeue: remove o elemento do inicio da fila
- front: consulta o elemento no inicio da fila sem removê-lo

Qual dessas estruturas você acha mais interessante? Se quiser, posso fornecer exemplos de código para ilustrar essas ideias!

Figura 11 – Resposta de pergunta sem contexto do Copilot sobre estrutura de dados

Array

Um array é uma estrutura de dados que armazena uma coleção de elementos, todos do mesmo tipo, em posições de memória contíguas. Os arrays têm tamanho fixo, determinado no momento da sua criação, e podem ser acessados diretamente através de índices.

Exemplo de um array:

```
Python Copiar
arr = [1, 2, 3, 4, 5]
print(arr[2]) # Saída: 3
```

Lista Ligada (Linked List)

Uma lista ligada é uma estrutura de dados composta por nós, onde cada nó contém um valor e uma referência (ou ponteiro) para o próximo nó na lista. Diferentemente dos arrays, as listas ligadas não têm tamanho fixo e os elementos não estão armazenados em posições de memória contíguas.

Exemplo de uma lista ligada em Python:

```
Python Copiar
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None

class LinkedList:
    def __init__(self):
        self.head = None

    def append(self, data):
        new_node = Node(data)
        if not self.head:
            self.head = new_node
            return
        last = self.head
        while last.next:
            last = last.next
        last.next = new_node
```

Diferenças principais:

- **Tamanho:** Arrays têm tamanho fixo; listas ligadas têm tamanho dinâmico.
- **Acesso:** Arrays permitem acesso direto via índice; listas ligadas requerem iteração através dos nós para acesso.
- **Memória:** Arrays ocupam memória contígua; listas ligadas não exigem contiguidade de memória.

Pilha (Stack)

Uma pilha é uma estrutura de dados que segue o princípio LIFO (Last In, First Out), onde o último elemento inserido é o primeiro a ser removido. As operações principais são "push" (inserir) e "pop" (remover).

Exemplo de uma pilha em Python:

```
Python Copiar
stack = []
stack.append(1) # push
stack.append(2)
print(stack.pop()) # Saída: 2
print(stack.pop()) # Saída: 1
```

Fila (Queue)

Uma fila é uma estrutura de dados que segue o princípio FIFO (First In, First Out), onde o primeiro elemento inserido é o primeiro a ser removido. As operações principais são "enqueue" (inserir) e "dequeue" (remover).

Exemplo de uma fila em Python:

```
Python Copiar
from collections import deque

queue = deque()
queue.append(1) # enqueue
queue.append(2)
print(queue.popleft()) # Saída: 1
print(queue.popleft()) # Saída: 2
```

Essas são explicações básicas e exemplos das estruturas de dados mencionadas. Precisa de mais detalhes?

Figura 12 – Resposta de pergunta com contexto do Copilot sobre estrutura de dados

perguntas individuais. A resposta da pergunta submetida sem contexto, forneceu explicações relativamente claras, apresentando conceitos de forma didática e com complexidade razoável para garantir uma compreensão. Quando o contexto foi adicionado, o Copilot expandiu ainda mais a resposta, oferecendo detalhes aprofundados e trechos inteiros de códigos, com exemplos práticos para ilustrar os conceitos apresentados. A inclusão dos códigos possibilitou uma resposta mais completa. Entretanto, isso impactou levemente a clareza, podendo gerar dificuldade para quem busca um entendimento mais superficial sobre determinado assunto ou mesmo para iniciantes.

Aprendizado Supervisionado vs. Não Supervisionado

- **Aprendizado Supervisionado:** Neste tipo de aprendizado, o algoritmo é treinado com um conjunto de dados rotulado, ou seja, os dados de entrada vêm com a resposta correta (rótulo). O objetivo é que o modelo aprenda a mapear entradas para saídas corretas, generalizando para novos dados. Exemplos clássicos incluem a classificação (como identificar e-mails spam) e a regressão (previsão de preços de casas).
- **Aprendizado Não Supervisionado:** Aqui, o algoritmo é treinado com dados que não possuem rótulos. O objetivo é encontrar padrões ou estruturas ocultas nos dados. Exemplos incluem agrupamento (clusterização), onde os dados são divididos em grupos com base em características similares, e a redução de dimensionalidade, que visa simplificar os dados mantendo o máximo de informação possível.

Algoritmo de Busca A*

O **algoritmo de busca A*** (pronuncia-se "A star") é um método usado em inteligência artificial e ciência da computação para encontrar o caminho mais curto entre dois pontos. Ele é amplamente utilizado em jogos, navegação de robôs e outros problemas de roteamento. O A* combina os pontos fortes de busca por largura e busca por custo uniforme usando uma função heurística para estimar o custo restante até o objetivo. Isso permite que ele encontre caminhos ótimos de maneira eficiente.

Aqui está uma visão simplificada de como ele funciona:

- **Função f(n):** Ele utiliza uma função $f(n) = g(n) + h(n)$, onde:
 - **g(n):** é o custo do caminho desde o ponto inicial até o ponto n.
 - **h(n):** é uma estimativa heurística do custo do ponto n até o objetivo.
- O A* expande os nós em ordem crescente de $f(n)$, garantindo que o nó com o menor custo total estimado seja explorado primeiro.

Essa combinação de custos garante que o A* seja tanto **completo** (encontra uma solução se ela existir) quanto **ótimo** (encontra a solução de menor custo).

Fascinante, não é? Há mais alguma área ou conceito dentro da inteligência artificial que você gostaria de explorar? 🤖💡

Figura 13 – Resposta de pergunta sem contexto do Copilot sobre IA

5.2 Resultados da Segunda Rodada de Testes

Nesta rodada, o intuito foi avaliar as três ferramentas com um foco mais direcionado à programação. De forma geral, os resultados foram semelhantes, porém levemente inferiores, envolvendo a área de elaboração de códigos a partir de algoritmos ou enunciados. As ferramentas apresentaram saídas que atenderam ao que foi solicitado.

Assim como na primeira rodada, chats foram criados com todas as perguntas agrupadas e individuais com e sem contexto, incluindo exemplos de execução para melhor compreensão. As mesmas perguntas foram feitas de forma idêntica nas três ferramentas, garantindo consistência nos testes.

Aprendizado Supervisionado vs. Não Supervisionado

Aprendizado Supervisionado

No aprendizado supervisionado, um modelo de machine learning é treinado utilizando um conjunto de dados rotulado, isso significa que os dados de entrada vêm acompanhados de suas respectivas respostas ou rótulos. O objetivo do modelo é aprender a mapear entradas para saídas corretas baseando-se nesses exemplos. O aprendizado supervisionado é comumente utilizado para tarefas de classificação e regressão.

Exemplos: Prever o preço de uma casa com base em características como localização, tamanho e número de quartos.

- Classificação:** Classificar e-mails como "spam" ou "não spam".
- Regressão:** Prever o valor de uma propriedade imobiliária com base em dados históricos.

```
Python Copiar
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

# Dados de entrada (features) e saída (target)
X = [[1, 2], [1, 3], [2, 1], [3, 1], [2, 2], [2, 3], [3, 2], [3, 3]]
y = [10000, 40000, 30000]

# Dividir os dados em conjuntos de treinamento e teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# Criar o modelo e treiná-lo
model = LinearRegression()
model.fit(X_train, y_train)

# Fazer previsão
previsao = model.predict(X_test)
```

Aprendizado Não Supervisionado

No aprendizado não supervisionado, o modelo é treinado utilizando um conjunto de dados que não possuem rótulos. O objetivo é identificar padrões ou estruturas ocultas nos dados. Esse tipo de aprendizado é frequentemente utilizado para tarefas de clustering e redução de dimensionalidade.

Exemplos: Agrupar clientes com comportamentos de compra semelhantes para campanhas de marketing direcionadas.

Clustering: Agrupar dados semelhantes em clusters (ex.: K-Means).

Redução de Dimensionalidade: Simplificar a representação dos dados (ex.: PCA - Análise de Componentes Principais).

Python Copiar

```
from sklearn.cluster import KMeans

# Dados de entrada (features)
X = [[1, 2], [2, 3], [2, 1], [3, 1], [2, 2], [2, 3], [3, 2], [3, 3]]

# Criar o modelo e treiná-lo
kmeans = KMeans(n_clusters=2)
kmeans.fit(X)

# Atribuir cada ponto a um cluster
Labels = kmeans.labels_
```

Algoritmo de Busca A*

O algoritmo de busca A* é um algoritmo de caminho mais curto utilizado em grafos. Ele é amplamente usado em aplicações que requerem a descoberta do caminho mais eficiente, como a navegação de robôs ou a busca de rotas em jogos. A* combina características da busca de custo uniforme e da busca gulosa, utilizando uma função de custo que considera tanto o custo do caminho desde o início quanto uma estimativa do custo restante até o objetivo.

Componentes principais:

- Custo Real ($g(n)$):** O custo do caminho desde o nó inicial até o nó n atual.
- Heurística ($h(n)$):** Uma estimativa do custo do nó n até o objetivo.
- Função de Avaliação ($f(n)$):** A soma do custo real e da heurística: $f(n) = g(n) + h(n)$.

Exemplo:

```
Python Copiar
import heapq

def heuristic(a, b):
    # Distância Manhattan (exemplo de heurística)
    return abs(a[0] - b[0]) + abs(a[1] - b[1])

def a_star(start, goal, neighbors):
    open_set = []
    heapq.heappush(open_set, (0, start))
    came_from = {}
    g_score = {start: 0}
    f_score = {start: heuristic(start, goal)}

    while open_set:
        _, current = heapq.heappop(open_set)

        if current == goal:
            path = []
            while current in came_from:
                path.append(current)
                current = came_from[current]
            path.append(goal)
            return path[::-1]

        for neighbor in neighbors(current):
            tentative_g_score = g_score[current] + 1 # 1 é a distância entre vizinhos
            if neighbor not in g_score or tentative_g_score < g_score[neighbor]:
                came_from[neighbor] = current
                g_score[neighbor] = tentative_g_score
                f_score[neighbor] = tentative_g_score + heuristic(neighbor, goal)
                heapq.heappush(open_set, (f_score[neighbor], neighbor))

    return None
```

Neste exemplo, `heuristic` calcula a distância estimada do nó atual ao objetivo, `a_star` executa o algoritmo A* para encontrar o caminho mais curto entre o nó inicial (`start`) e o objetivo (`goal`), considerando os vizinhos do nó atual.

Essas são as explicações sobre aprendizado supervisionado e não supervisionado, além do algoritmo de busca A*. Espero que sejam úteis! Se precisar de mais detalhes ou tiver outras perguntas, estou à disposição.

Figura 14 – Resposta de pergunta com contexto do Copilot sobre IA

O ChatGPT teve um desempenho menor, em alguns casos até inferior às outras ferramentas. O Gemini teve um ponto relevante de diversidade de modelos de resposta, com conteúdos variando entre conceitos e muitas tabelas comparativas ajudando na compreensão geral. O Copilot teve um desempenho interessante, com respostas mais bem estruturadas e profundas, com códigos concisos e funcionais.

5.2.1 ChatGPT

A figura 15 dos critérios de testes do ChatGPT revela que o modelo individual com e sem contexto tiveram uma grande diferença em relação aos agrupados. inclusive no requisito de relevância, que na primeira rodada estavam bem equilibrados, nessa rodada teve uma diferença considerável.

As respostas individuais com contexto apresentaram a maior pontuação em praticamente todos os requisitos, mostrando que o ChatGPT para os casos de perguntas sobre programação funciona bem melhor com contexto e de forma individual. As respostas individuais sem contexto, mesmo não tendo alcançado a mesma profundidade, ainda tiveram uma boa performance, especialmente no desvio do assunto e corretude. Por outro lado, as respostas agrupadas com contexto tem uma melhor resolução em comparação com as respostas agrupadas sem contexto em critérios como corretude, complexidade, proximidade com a resposta esperada, profundidade e relevância. No entanto, o modelo sem contexto se destacou em termos de clareza e desvio no assunto no âmbito de agrupados, mostrando que, em casos onde as perguntas são apresentadas de forma agrupada, a adição de contexto pode dificultar a objetividade das respostas em certos momentos. De maneira geral, os testes destacam que a contextualização e a forma de envio das perguntas são fatores essenciais para a garantia de respostas mais completas e profundas.

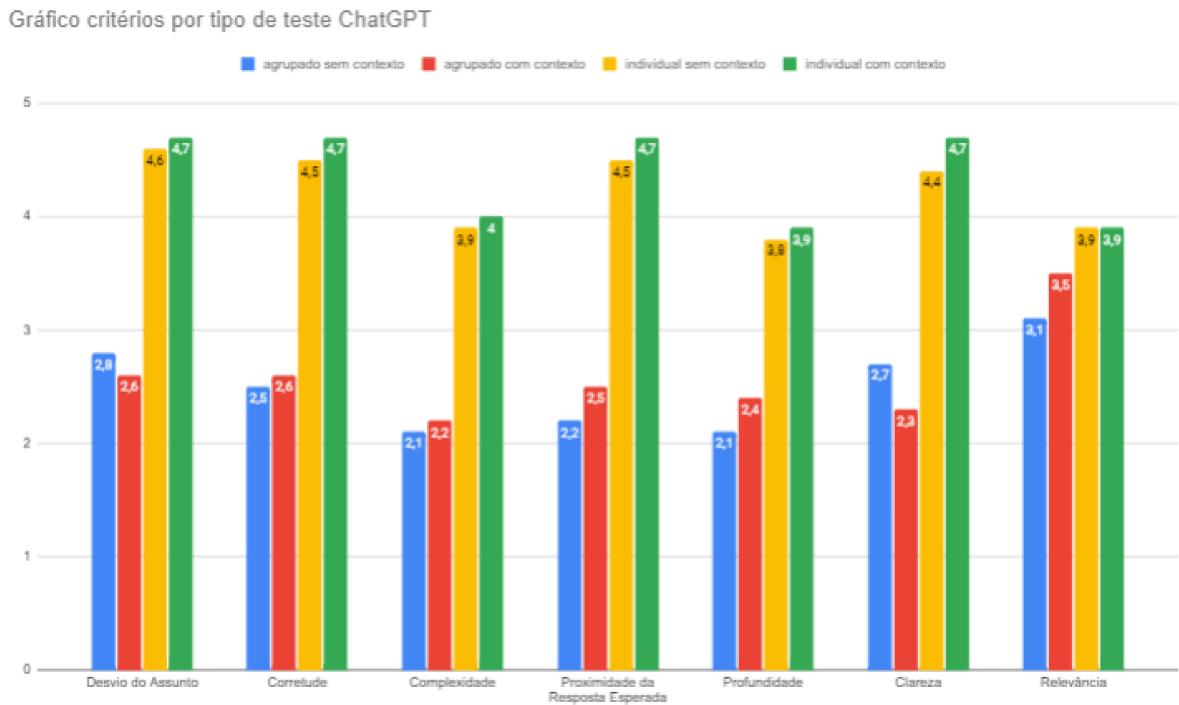


Figura 15 – Figura do ChatGPT de critérios por tipo de teste

Um exemplo da eficácia nas respostas do ChatGPT pode ser observado na análise da questão sobre algoritmos de ordenação, o mergesort. A comparação entre as respostas agrupadas e individuais, ambas com contexto, evidencia uma diferença significativa, conforme mostrado nas imagens 16 (agrupadas) e 17 (individuais). Na resposta agrupada, o conteúdo foi apresentado de forma breve, com o código completo, porém sem explicações ou exemplos de execução solicitados, o que demonstra que a apresentação de perguntas de forma agrupada pode impactar a completude da resposta, resultando em omissões de informações importantes. Por outro lado, a resposta individual com contexto foi muito mais detalhada, incluindo a definição do algoritmo, o código comentado, explicações claras, um exemplo de execução e uma descrição do funcionamento, além de um resumo conclusivo. Isso ressalta a importância do formato individual com contexto para obter respostas mais abrangentes e informativas.

5.2.2 Gemini

A figura 18 dos critérios de teste do Gemini revela uma competição acirrada entre os formatos individuais com e sem contexto, que se destacam como os mais bem avaliados em todos os critérios. Ambos atingiram a pontuação máxima de 5 nos aspectos de desvio do assunto e corretude. Nos critérios de complexidade, proximidade da resposta esperada, profundidade e clareza, o formato individual sem contexto obteve os melhores resultados, sugerindo que, para casos específicos de exercícios de programação, a inclusão de contexto

10. Algoritmos de Ordenação

Pergunta: Implemente o algoritmo de ordenação Merge Sort para ordenar uma lista de números fornecida pelo usuário e mostre um exemplo de execução.

Resposta:

```
python Copy code
```

```
# Implementação do Merge Sort
def merge_sort(lista):
    if len(lista) > 1:
        meio = len(lista) // 2
        esquerda = lista[:meio]
        direita = lista[meio:]

        merge_sort(esquerda)
        merge_sort(direita)

        i = j = k = 0
        while i < len(esquerda) and j < len(direita):
            if esquerda[i] < direita[j]:
                lista[k] = esquerda[i]
                i += 1
            else:
                lista[k] = direita[j]
                j += 1
            k += 1

        while i < len(esquerda):
            lista[k] = esquerda[i]
            i += 1
            k += 1

        while j < len(direita):
            lista[k] = direita[j]
            j += 1
            k += 1

# Exemplo de execução
numeros = [38, 27, 43, 3, 9, 82, 10]
merge_sort(numeros)
print("Lista ordenada:", numeros)
```

Esses exemplos atendem aos critérios de clareza e precisão, com explicações e execuções práticas.

Figura 16 – Resposta de pergunta agrupadas do ChatGPT sobre mergesort

da forma testada pode, em algumas situações, impactar negativamente o desempenho. O modelo agrupado com contexto também apresentou um desempenho satisfatório, especialmente em complexidade e corretude, embora não alcance o nível de excelência das respostas individuais. Já o modelo agrupado sem contexto demonstrou resultados inferiores, com as menores pontuações em complexidade, proximidade da resposta esperada e profundidade, indicando que esse formato é menos eficaz. Por outro lado, o critério de relevância mostrou variações menores entre os modelos, com resultados mais consistentes.

O Gemini teve um caso de resposta inesperado, na resolução da pergunta de variáveis e operações aritméticas nos testes individuais sem contexto (figura 19), ele trouxe todo o conteúdo bem completo e cheio de explicações, inclusive até o pseudocódigo, o código em python com a explicação, exemplo de saída, porque usar o tipo float, que é uma informação adicional, e além de tudo isso, ele trouxe um diferencial de alguns exercícios propostos para fazer baseado na pergunta, foi um ponto bem interessante para quem está

Algoritmos de Ordenação - Merge Sort

O Merge Sort é um algoritmo de ordenação baseado na técnica de dividir e conquistar. Ele divide recursivamente uma lista em duas metades, ordena cada metade e, em seguida, junta (merge) as duas metades ordenadas. A complexidade desse algoritmo é $O(n \log n)$, o que o torna eficiente para listas grandes.

Explaination do Merge Sort:

- Dividir: A lista é dividida recursivamente até que cada sublista tenha apenas um elemento.
- Conquistar: Durante a fusão, as sublistas são ordenadas e combinadas de forma a formar uma lista ordenada.
- Fusão: Duas listas ordenadas são combinadas em uma lista maior, garantindo que a ordem seja mantida.

Código em Python para Merge Sort:

```
python
# Função para realizar a fusão de duas listas ordenadas
def merge(list1, list2):
    resultado = []
    i, j = 0, 0

    # Enquanto ambos os indicadores forem válidos
    while i < len(list1) and j < len(list2):
        if list1[i] < list2[j]:
            resultado.append(list1[i])
            i += 1
        else:
            resultado.append(list2[j])
            j += 1

    # Adiciona o restante dos elementos
    resultado.extend(list1[i:])
    resultado.extend(list2[j:])

    return resultado

# Função principal do Merge Sort
def merge_sort(lista):
    if len(lista) <= 1:
        return lista

    # Divide a lista ao meio
    meio = len(lista) // 2
    lista1 = lista[:meio]
    lista2 = lista[meio:]

    # Recursão para dividir e ordenar as sublistas
    lista1 = merge_sort(lista1)
    lista2 = merge_sort(lista2)

    # Fusão das sublistas ordenadas
    return merge(lista1, lista2)

# Função para ler a lista de números do usuário
def main():
    # Lê os números do usuário e os converte para uma lista de inteiros
    entrada = input("Digite uma lista de números separados por espaço: ")
    lista = list(map(int, entrada.split()))

    # Ordena a lista usando o Merge Sort
    lista_ordenada = merge_sort(lista)

    # Exibe a lista ordenada
    print("Lista ordenada: ", lista_ordenada)

    # Chama a função principal
    main()

# Explicação do Código:
1. Função merge(list1, list2):
    • Recebe duas listas já ordenadas (list1 e list2).
    • Funde as duas listas em uma única lista ordenada comparando os elementos da esquerda (list1) e da direita (list2).
    • A função retorna a lista resultante do fusão.

2. Função merge_sort(lista):
    • Se a lista tiver apenas um elemento ou estiver vazia, já está ordenada, então retorna a lista.
    • Caso contrário, a lista é dividida ao meio e a função é chamada recursivamente para ordenar as duas metades.
    • Por fim, as duas metades ordenadas são combinadas (fundidas) usando a função merge.

3. Função main():
    • Solicita ao usuário que forneça uma lista de números separados por espaço.
    • A entrada é convertida para uma lista de inteiros.
    • A lista é ordenada utilizando o algoritmo merge_sort() e o resultado é impresso.

# Saída esperada:
Entrada:
Lista ordenada: [1, 7, 23, 32, 5, 62, 32]

Como o Merge Sort Funciona:
1. Dividir:
    • A lista original [14, 7, 23, 32, 5, 62, 32] é dividida recursivamente até chegar em listas de um único elemento.
2. Fusão:
    • As listas menores são combinadas e ordenadas durante a fusão. Por exemplo, [7] e [32] se tornam [7, 32].
3. Recursão:
    • A recursão é repetida para cada sublista, até que toda a lista esteja ordenada.

Resumo:
O Merge Sort é um algoritmo de ordenação eficiente com complexidade  $O(n \log n)$ . Ele funciona dividindo recursivamente a lista e depois fundindo as partes ordenadas. O código implementado lê uma lista de números fornecida pelo usuário, aplica o Merge Sort e exibe a lista ordenada.

Copy code
```

Saída esperada:

Entrada:

Como o Merge Sort Funciona:

- Dividir:**
 - A lista original [14, 7, 23, 32, 5, 62, 32] é dividida recursivamente até chegar em listas de um único elemento.
- Fusão:**
 - As listas menores são combinadas e ordenadas durante a fusão. Por exemplo, [7] e [32] se tornam [7, 32].
- Recursão:**
 - A recursão é repetida para cada sublista, até que toda a lista esteja ordenada.

Resumo:

O Merge Sort é um algoritmo de ordenação eficiente com complexidade $O(n \log n)$. Ele funciona dividindo recursivamente a lista e depois fundindo as partes ordenadas. O código implementado lê uma lista de números fornecida pelo usuário, aplica o Merge Sort e exibe a lista ordenada.

Figura 17 – Resposta de pergunta individual do ChatGPT sobre mergesort

Gráfico critérios por tipo de teste Gemini

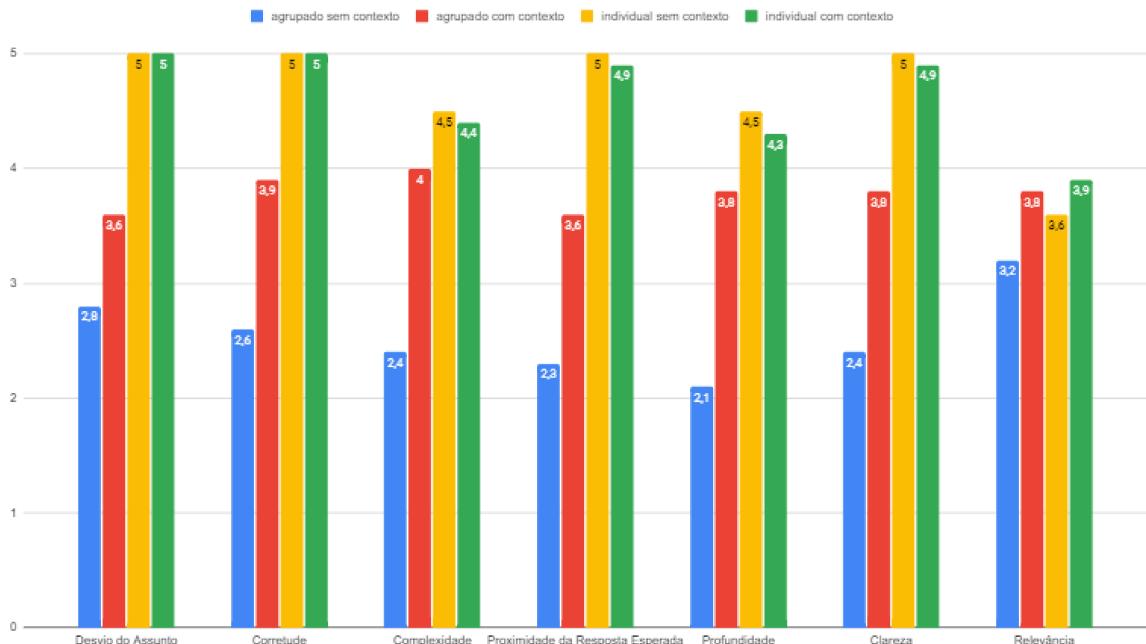


Figura 18 – Figura do Gemini de critérios por tipo de teste

iniciando na programação.

<p>Variáveis, Operações Aritméticas e Entrada do Usuário em Python</p> <p>Entendendo o Problema:</p> <p>Queremos criar um programa que:</p> <ol style="list-style-type: none"> 1. Peça ao usuário para digitar dois números. 2. Armazene esses números em variáveis. 3. Calcule a soma desses números. 4. Exiba o resultado na tela. <p>Solução em Python:</p> <pre style="background-color: #f0f0f0; padding: 5px;">Python # Pedimos ao usuário para digitar os dois números numero1 = float(input("Digite o primeiro número: ")) numero2 = float(input("Digite o segundo número: ")) # Calculamos a soma soma = numero1 + numero2 # Exibimos o resultado print("A soma dos números é:", soma)</pre> <p>Use o código com cuidado.</p> <p>Explicando o Código:</p> <ul style="list-style-type: none"> 1. <code>float(input("digite o primeiro número: "))</code>: <ul style="list-style-type: none"> • <code>input()</code> permite que o usuário digite um valor e o programa o leia como texto. • <code>float()</code> converte esse texto para um número de ponto flutuante (um número com casas decimais), para que possamos realizar operações matemáticas. • O texto dentro das aspas é a mensagem que aparece para o usuário, pedindo que ele digite o número. 2. <code>numero1 = ... e numero2 = ...</code>: <ul style="list-style-type: none"> • Os valores obtidos da entrada do usuário são armazenados nas variáveis <code>numero1</code> e <code>numero2</code>, respectivamente. 3. <code>soma = numero1 + numero2</code>: <ul style="list-style-type: none"> • A operação de soma é realizada e o resultado é armazenado na variável <code>soma</code>. 4. <code>print("A soma dos números é:", soma)</code>: <ul style="list-style-type: none"> • A função <code>print()</code> exibe a mensagem dentro das aspas e o valor da variável <code>soma</code> na tela. <p>Exemplo de Execução:</p>	<p>Se o usuário digitar <code>5</code> para o primeiro número e <code>3.14</code> para o segundo, a saída será:</p> <pre style="background-color: #f0f0f0; padding: 5px;">Digite o primeiro número: 5 Digite o segundo número: 3.14 A soma dos números é: 8.14</pre> <p>Por que usar <code>float</code>?</p> <ul style="list-style-type: none"> • Números com casas decimais: O tipo <code>float</code> permite trabalhar com números que possuem parte decimal, como <code>3.14</code>. • Versatilidade: Se o usuário digitar um número inteiro (sem casas decimais), o Python automaticamente o converte para um <code>float</code>. <p>Observações:</p> <ul style="list-style-type: none"> • Outras operações: Além da soma, você pode realizar outras operações aritméticas como subtração (<code>-</code>), multiplicação (<code>*</code>) e divisão (<code>/</code>). • Tipos de dados: Python possui outros tipos de dados além de <code>float</code>, como <code>int</code> para números inteiros e <code>str</code> para textos. • Variáveis: As variáveis são como caixas onde você armazena dados. O nome da variável pode ser qualquer palavra, desde que siga algumas regras. <p>Exercícios:</p> <ul style="list-style-type: none"> • Modifique o programa para calcular a diferença, o produto e a divisão dos dois números. • Peça ao usuário para digitar três números e calcule a média. • Crie um programa que converta temperatura de Celsius para Fahrenheit. <p>Este é apenas um exemplo básico. A programação oferece infinitas possibilidades!</p>
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Figura 19 – resposta de pergunta do Gemini para o teste de variáveis e operações

5.2.3 Copilot

A figura de critérios do Copilot (figura 20) revela que os testes individuais são mais avaliados em elaboração de respostas, com destaque para as respostas individuais com contexto, que se destacaram mais no critério de corretude, atingindo 100% de acerto, e também se destacou em complexidade e desvio do assunto, demonstrando uma capacidade de fornecer respostas detalhadas. Os testes individuais sem contexto também apresentaram resultados positivos, destacando-se em clareza, relevância e proximidade com a resposta esperada. Isso mostra que, em alguns casos, respostas sem contexto podem oferecer melhor acessibilidade e compreensão do conteúdo.

Por outro lado, os testes agrupados tiveram um desempenho bem inferior, com pontuações baixas na maioria dos critérios. A relevância foi o único critério em que apresentaram uma performance um pouco mais elevada. Esses resultados indicam que, para perguntas relacionadas à programação, o Copilot responde de forma mais eficaz quando as questões são enviadas individualmente, pois o agrupamento de perguntas parece prejudicar a qualidade das respostas.

Um exemplo que ilustra a capacidade do Copilot em responder a perguntas relacionadas à programação está nas imagens 21 e 22, que apresenta a resposta para a pergunta sobre listas. Nas respostas individuais com contexto, o Copilot frequentemente inclui cód

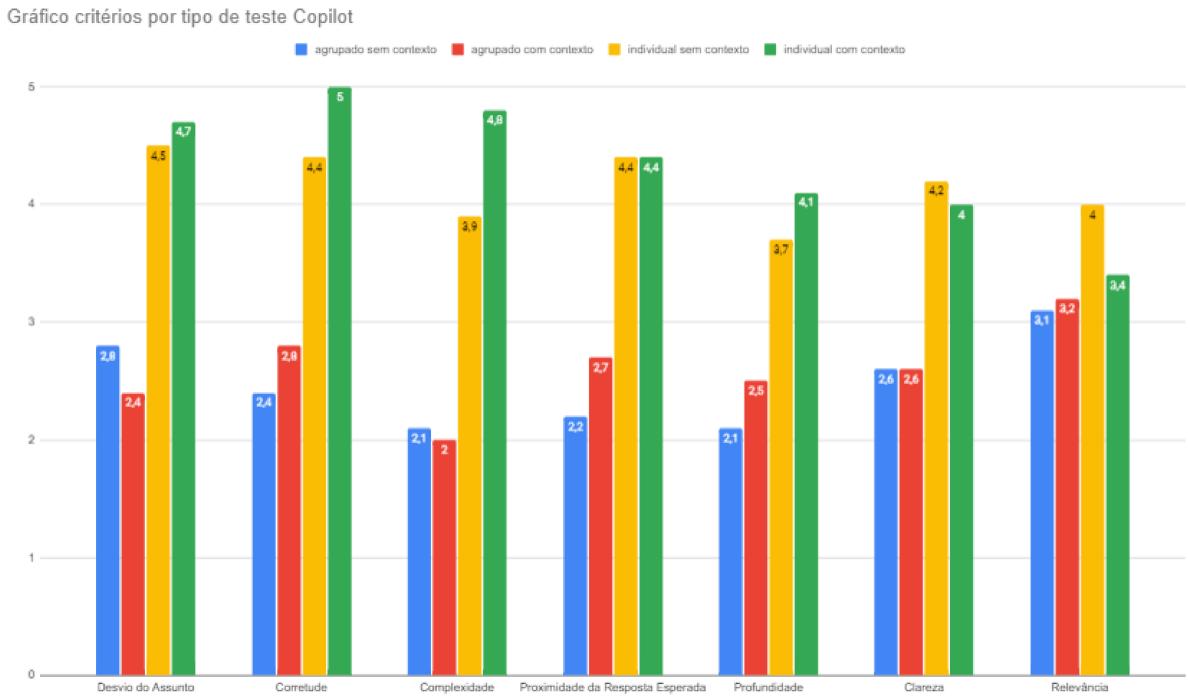


Figura 20 – Figura do Copilot de critérios por tipo de teste

digos em diversas linguagens de programação, como Python, C, Java e, ocasionalmente, JavaScript, com explicações detalhadas de cada trecho. No caso específico das listas, ele manteve esse padrão, oferecendo uma resposta rica em detalhes, com pontos informativos relevantes e exemplos práticos de execução, além dos códigos completos acompanhados de explicações claras. Essa abordagem reforça a versatilidade da ferramenta em apresentar soluções abrangentes e de fácil compreensão.

5.3 Comparação de Resultados entre Primeira e Segunda Rodada

A figura 23 apresentado oferece uma visão comparativa do desempenho dos três modelos de IA (ChatGPT, Gemini e Copilot) nas duas rodadas de avaliação. Ele foi gerado a partir da média das pontuações coletadas de cada teste e organizadas por tipo de ferramenta e os critérios apresentados.

Na primeira rodada, o ChatGPT se destacou em todos os critérios, tendo apenas um empate em relevância com o Copilot. O Gemini e o Copilot tiveram resultados parecidos, porém o Copilot teve vantagem sobre ele, contando com superioridade em relação ao Gemini em desvio do assunto, corretude, clareza e relevância, com diferenças mínimas. Já o Gemini ganhou do Copilot em profundidade e empatou em proximidade com a resposta esperada.

Na segunda rodada, observa-se uma leve diminuição no desempenho geral dos mo-

Listas

Vamos escrever um programa que pede ao usuário para inserir 5 números, armazena esses números em uma lista e, em seguida, exibe a lista na tela. Vou explicar cada linha de código e mostrar exemplos em Python, Java e C.

Python:

```
Python Copiar
# Cria uma lista vazia para armazenar os números
numeros = []

# Solicita 5 números ao usuário e armazena-os na lista
for i in range(5):
    numero = float(input("Digite o número {}: ".format(i+1)))
    numeros.append(numero)

# Exibe a lista de números
print("Os números digitados são:", numeros)
```

Explicação linha por linha:

1. `numeros = []` : Cria uma lista vazia chamada `numeros` para armazenar os números digitados pelo usuário.
2. `for i in range(5)` : Inicia um laço `for` que será executado 5 vezes.
3. `numero = float(input("Digite o número {}: ".format(i+1)))` : Solicita ao usuário para digitar um número e converte-o para o tipo `float`.
4. `numeros.append(numero)` : Adiciona o número digitado à lista `numeros`.
5. `print("Os números digitados são:", numeros)` : Exibe a lista de números digitados.

Exemplo de execução:

```
Copiar
Digite o número 1: 10.5
Digite o número 2: 20.3
Digite o número 3: 30.7
Digite o número 4: 40.1
Digite o número 5: 50.0
Os números digitados são: [10.5, 20.3, 30.7, 40.1, 50.0]
```

Java:

```
Java Copiar
import java.util.ArrayList;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        ArrayList<Double> numeros = new ArrayList<>();

        // Solicita 5 números ao usuário e armazena-os na lista
        for (int i = 0; i < 5; i++) {
            System.out.print("Digite o número " + (i + 1) + ": ");
            double numero = scanner.nextDouble();
            numeros.add(numero);
        }

        // Exibe a lista de números
        System.out.println("Os números digitados são: " + numeros);
    }
}
```

Explicação linha por linha:

1. `Scanner scanner = new Scanner(System.in);` : Cria um objeto `Scanner` para ler a entrada do usuário.
2. `ArrayList<Double> numeros = new ArrayList<>();` : Cria uma lista vazia chamada `numeros` para armazenar os números digitados pelo usuário.
3. `for (int i = 0; i < 5; i++) { ... }` : Inicia um laço `for` que será executado 5 vezes.
4. `System.out.print("Digite o número " + (i + 1) + ": ");` : Solicita ao usuário para digitar um número.
5. `double numero = scanner.nextDouble();` : Lê o número digitado pelo usuário e converte-o para o tipo `double`.
6. `numeros.add(numero);` : Adiciona o número digitado à lista `numeros`.
7. `System.out.println("Os números digitados são: " + numeros);` : Exibe a lista de números digitados.

Figura 21 – Resposta de pergunta do Copilot para o perguntas de listas parte 1

Exemplo de execução:

```
Copiar
Digite o número 1: 10.5
Digite o número 2: 20.3
Digite o número 3: 30.7
Digite o número 4: 40.1
Digite o número 5: 50.0
Os números digitados são: [10.5, 20.3, 30.7, 40.1, 50.0]
```

C:

```
C Copiar
#include <stdio.h>

int main() {
    double numeros[5];

    // Solicita 5 números ao usuário e armazena-os na lista
    for (int i = 0; i < 5; i++) {
        printf("Digite o número %d: ", i + 1);
        scanf("%lf", &numeros[i]);
    }

    // Exibe a lista de números
    printf("Os números digitados são: ");
    for (int i = 0; i < 5; i++) {
        printf("%.2lf ", numeros[i]);
    }
    printf("\n");
}

return 0;
```

Explicação linha por linha:

1. `double numeros[5];` : Declara um array de 5 elementos do tipo `double` para armazenar os números digitados pelo usuário.
2. `for (int i = 0; i < 5; i++) { ... }` : Inicia um laço `for` que será executado 5 vezes.
3. `printf("Digite o número %d: ", i + 1);` : Solicita ao usuário para digitar um número.
4. `scanf("%lf", &numeros[i]);` : Lê o número digitado pelo usuário e armazena-o no array `numeros`.
5. `printf("Os números digitados são: ");` : Prepara para exibir a lista de números digitados.
6. `for (int i = 0; i < 5; i++) { ... }` : Inicia um laço `for` para exibir cada número no array `numeros`.
7. `printf("%.2lf ", numeros[i]);` : Exibe cada número do array `numeros`.
8. `printf("\n");` : Adiciona uma nova linha após exibir todos os números.

Exemplo de execução:

```
Copiar
Digite o número 1: 10.5
Digite o número 2: 20.3
Digite o número 3: 30.7
Digite o número 4: 40.1
Digite o número 5: 50.0
Os números digitados são: 10.50 20.30 30.70 40.10 50.00
```

Espero que esses exemplos ajudem! Se tiver mais alguma pergunta ou precisar de mais exemplos, estou aqui para ajudar.

Figura 22 – Resposta de pergunta do Copilot para o perguntas de listas parte 2

deles, com exceção do Copilot que apresentou uma pequena melhora em alguns critérios, como em complexidade que obteve melhor resultado. O ChatGPT teve uma queda em relação à rodada anterior, empatando com o Gemini e em alguns critérios como complexidade e relevância e sendo melhor em apenas profundidade e desvio do assunto por muito pouco com o Copilot. O Gemini teve melhor resultado em corretude e proximidade com a resposta esperada, não indo tão bem em desvio do assunto, profundidade e clareza.

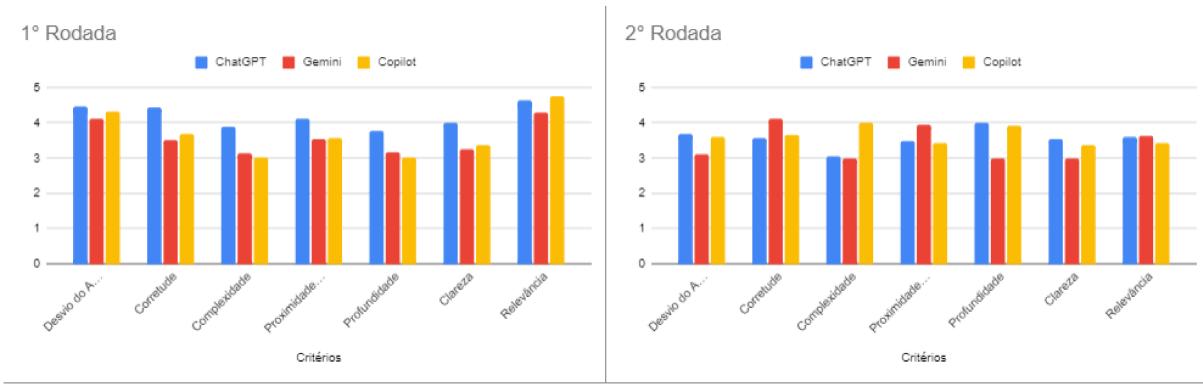


Figura 23 – Figura de Comparação entre Primeira e Segunda rodada

5.4 Resultados de Testes Especiais

Os testes feitos nesta seção foram baseados em pré-configurações específicas disponibilizadas pela ferramenta. Somente o ChatGPT dispunha deste tipo de funcionalidade até a data de execução dos testes para esta monografia. Também foi abordado os testes relacionados a submissão de PDFs, contendo algumas informações e solicitando algumas informações, como melhores pontos, geração de perguntas e validação de respostas. A seção 5.4.1 traz informações dos testes de personalização e a seção 5.4.2 os testes com arquivos de dados

5.4.1 Testes de Personalização

Os testes de personalização apresentaram resultados mais parecidos com os testes feitos nas perguntas com contextualização. Para a personalização, foi adicionado o perfil de uma jovem estudante de Ciência da Computação que trabalha especificamente na área de DevOps. Também foi configurado para que as respostas fossem claras, bem definidas, de fácil entendimento, e incluíssem exemplos sempre que possível.

Após os testes, a maioria das respostas apresentou uma maior quantidade de trechos de código, como demonstrado nos exemplos 24 e 30. No primeiro exemplo 24, ele representa a resposta do ChatGPT em um teste individual com contexto, os resultados foram satisfatórios, mas menos detalhados. Na figura 30, que mostra o resultado com a personalização configurada, observou-se uma maior inserção de códigos, além da variação das linguagens de programação utilizadas. De forma geral, a inclusão da personalização no ChatGPT resultou em respostas mais focadas e direcionadas, enriquecendo o conteúdo com aspectos específicos do ramo configurado e aprimorando a experiência de aprendizado.



Figura 24 – Resposta de pergunta individual com contexto do ChatGPT sobre algoritmo

5.4.2 Testes com Arquivos de Dados

Os testes com alimentação de dados foram realizados em três diferentes modalidades de abordagem: Respostas Relacionadas a Dúvidas Gerais, Geração de conteúdo a Partir de PDF e Validação de Questões sobre Programação. Esses tópicos trazem a divisão dos testes feitos com o uso de PDFs.

Respostas Relacionadas a Dúvidas Gerais

Um dos testes envolveu um artigo sobre a criação de um jogo chamado Layer-Network. Um jogo de código aberto voltado para o ensino interativo da disciplina de redes de computadores, com ênfase nas camadas de rede Reis e Souza (2020). Com o artigo em mãos, antes de submetê-lo à IA, foi perguntado ao ChatGPT a seguinte frase: 'Me fale sobre o jogo LayerNetwork'. No entanto, a resposta fornecida foi sobre LayerNet, uma plataforma de jogos no Telegram, que não tem relação com o artigo, o que demonstra uma confusão entre os termos e uma limitação no reconhecimento de informações novas por parte da IA. A resposta foi conforme a figura 25.

Após essa etapa, foi submetido o artigo à IA e feito uma pergunta semelhante: 'Baseado neste artigo, me fale sobre o jogo LayerNetwork'. Desta vez, a IA descreveu com precisão o que é o jogo, bem informativo, detalhando sua estrutura, o funcionamento, como é a jogabilidade, a metodologia utilizada com as linguagens e ferramentas gerais, como foi criado e seu objetivo educacional de trazer o conteúdo mais denso de uma forma mais interativa, a figura 26 mostra isso.

Um outro teste consistiu em fornecer ao ChatGPT um PDF contendo a ficha de uma disciplina acadêmica, com informações abrangentes sobre o panorama geral da matéria, incluindo o conteúdo programático, objetivos e cronograma. A pergunta feita ao



Figura 25 – Resposta de pergunta no ChatGPT sobre jogo LayerNetwork antes da submissão do PDF

chat foi para que ele revelasse os principais pontos e gerasse uma lista com os tópicos da disciplina. Enunciado submetido:

"Me fale sobre os principais pontos dessa disciplina, uma lista do conteúdo que será ministrado"

A resposta foi bastante objetiva, atendendo exatamente ao que foi solicitado. Com base no PDF fornecido, a IA conseguiu extrair o conteúdo e apresentá-lo de maneira clara e concisa. Além disso, o processamento foi surpreendentemente rápido, considerando que o documento tinha 27 páginas. A capacidade de responder praticamente em tempo real foi impressionante, a resposta se encontra na figura 27.

Geração de Conteúdo a Partir de PDF

Essa etapa foi realizada utilizando algumas pré-configurações. Primeiro, foi criado um PDF simples contendo a definição do que é programação. O conteúdo era o seguinte:

"Programação é o processo de escrita, teste e manutenção de um programa de computador. O programa é escrito em uma linguagem de programação, embora seja possível, com alguma dificuldade, escrevê-lo diretamente em linguagem de máquina. Diferentes partes de um programa podem ser escritas em diferentes linguagens, e essas linguagens funcionam de maneiras distintas. Por esse motivo, programadores podem desenvolver programas muito diferentes para linguagens diferentes, ainda que, teoricamente, a maioria das linguagens possa ser usada para criar qualquer programa."

The screenshot shows a ChatGPT interface. At the top right, there is a download icon labeled 'artigo-testetcc.pdf PDF'. Below it, a message says 'baseado nesse artigo me fale sobre o jogo LayerNetwork'. The main content is a response from a user (indicated by a profile picture) about the game LayerNetwork. The response discusses its educational purpose, structure, methodology, and educational objective. It also mentions its interactive nature and how it combines theoretical knowledge with practical simulations. At the bottom of the message, there are several small icons for interacting with the AI.

O jogo LayerNetwork, conforme descrito no artigo que você forneceu, é um jogo educacional desenvolvido com o objetivo de auxiliar no processo de ensino-aprendizagem das camadas de redes de computadores. Ele foi projetado para integrar teoria e prática, proporcionando aos alunos uma melhor compreensão dos conteúdos relacionados a redes de computadores, especialmente das camadas de rede.

Estrutura do Jogo

O enredo do LayerNetwork faz uma analogia com o processo de postagem de encomendas nos correios, utilizando personagens do filme Matrix, como Neo e Morpheus. Durante o jogo, o jogador assume o papel de Neo e deve completar as etapas relacionadas às camadas de redes, como a camada de aplicação, camada de transporte, camada de rede, camada de enlace e camada física, simulando a entrega de pacotes de dados de uma maneira lúdica e envolvente (artigo-testetcc).

Metodologia

O jogo foi desenvolvido utilizando a engine Unity3D e a linguagem de programação C#, aproveitando as vantagens dessa ferramenta para criar um ambiente interativo tanto para desktop quanto para Android. Durante o jogo, o aluno aplica conhecimentos teóricos sobre redes de computadores de forma prática, facilitando a fixação do conteúdo estudado (artigo-testetcc).

Objetivo Educacional

O foco do LayerNetwork é permitir que os alunos pratiquem e consolidem conceitos como protocolos de redes, roteamento de pacotes e controle de fluxo, conforme progridem pelas camadas do modelo de referência. O jogo oferece uma experiência interativa e desafiante, com perguntas, desafios e análises de rotas de rede, permitindo que o aluno aplique seus conhecimentos teóricos de forma prática e divertida (artigo-testetcc).

Em resumo, o LayerNetwork é uma ferramenta educacional poderosa, combinando aprendizado teórico com simulações práticas e interativas, visando melhorar a compreensão das redes de computadores.

Figura 26 – Resposta do ChatGPT sobre jogo LayerNetwork apos submissão de pdf

Foi submetido o PDF ao ChatGPT, solicitando que gerasse algumas perguntas com base no texto fornecido. O resultado foi bastante satisfatório, pois ele conseguiu criar perguntas relevantes e, em alguns casos, com um nível de dificuldade interessante e desafiador. Isso demonstra a capacidade das IA de compreender o conteúdo e formular questionamentos úteis para o aprendizado. Na figura 28 mostra alguns exemplos de perguntas geradas a partir do texto no ChatGPT em cima do conteúdo.

Essas perguntas no ChatGPT não apenas capturam a essência do conteúdo, mas também incentivam o estudante a refletir sobre aspectos mais profundos da programação, como os detalhes entre diferentes linguagens e o impacto que isso pode ter no desenvolvimento de software. Esse tipo de questionamento é valioso tanto para iniciantes quanto

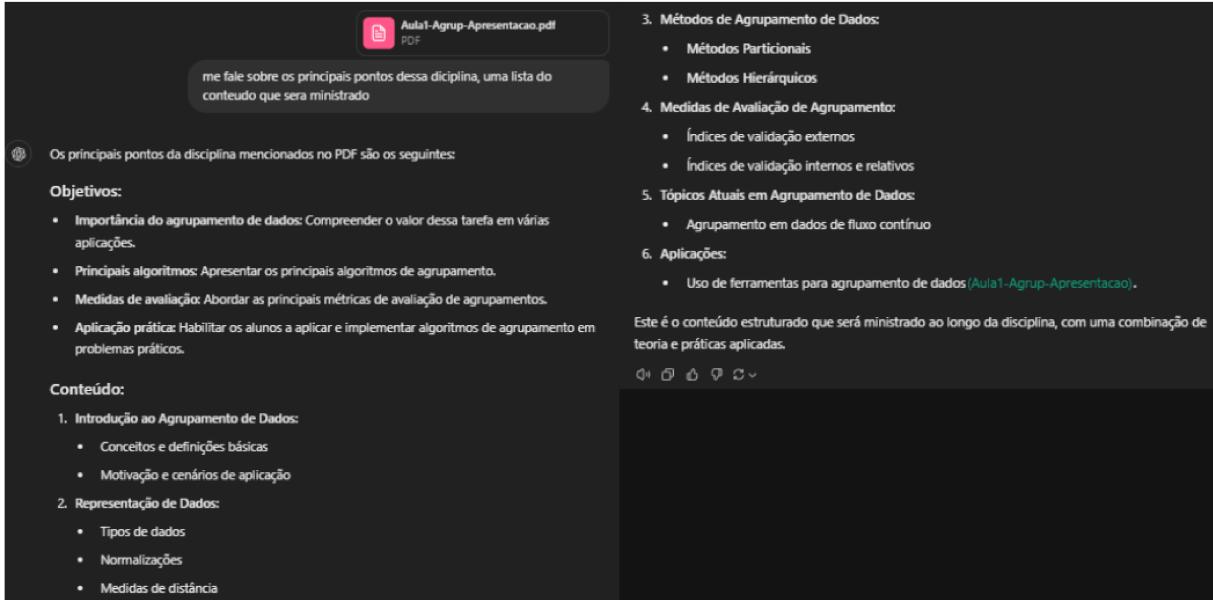


Figura 27 – Resposta do ChatGPT sobre ficha de disciplina com submissão de pdf

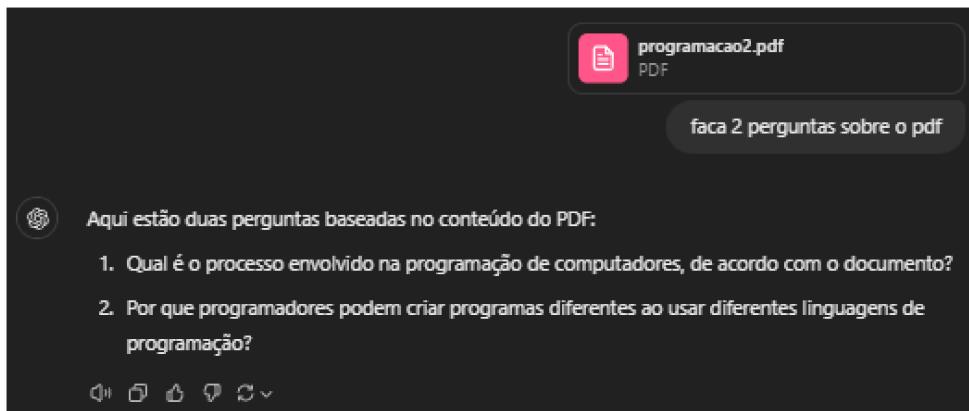


Figura 28 – Resposta de pergunta do ChatGPT sobre geração de perguntas com submissão de pdf

para programadores mais experientes, pois promove uma compreensão mais abrangente do tema.

Validação de Questões sobre Programação

Os testes de validação de questões têm a função de realizar uma análise crítica baseada no conteúdo submetido, comparando com o modelo de resposta já submetido. Nesse processo, foi disponibilizado um PDF com uma pergunta relacionada a um conteúdo de programação e contendo também a resposta esperada. A partir disso, foi submetido outro PDF tendo uma resposta um pouco diferente do PDF que já tinha sido enviado. Com isso, foi solicitado a comparação entre a resposta enviada depois, com o modelo de resposta esperada para o ChatGPT. Com base nessa análise, foi gerado um relatório contendo a porcentagem de acertos e sugestões de possíveis áreas de estudo para aprimoramento.

Um PDF inicial com o modelo de resposta esperada foi submetido, acompanhado com o enunciado para instruir a IA sobre o salvamento do conteúdo do PDF: 'guarde esse pdf, vou mandar a seguir algo relacionado a ele'. O conteúdo do pdf foi o seguinte:

Pergunta 1: Escreva um programa que peça ao usuário dois números e exiba a soma deles.

- Resposta Esperada (Python):

```
num1 = float(input("Digite o primeiro número: "))
num2 = float(input("Digite o segundo número: "))
soma = num1 + num2
print("A soma é:", soma)
```

Após essa instrução, o segundo PDF com a resposta alterada foi submetido, seguindo praticamente o mesmo modelo, alterando alguns pontos como o texto e o nome das variáveis. Também foi acompanhado do seguinte enunciado: 'o aluno respondeu esse conteúdo do pdf, gostaria de dar um retorno se ele acertou, qual a porcentagem de acerto e caso tenha algo que possa melhorar me mande uma recomendação'. O conteúdo foi o seguinte código:

```
n1 = float(input("Número 1: "))
n2 = float(input("Número 2: "))
soma = n1 + n2
print("A soma é:", soma)
```

A resposta do ChatGPT foi apresentada na figura 29. Ele fez uma análise correta, bem detalhada, com o relatório e apresentou uma resposta profunda. A resolução do ChatGPT atendeu aos requisitos solicitados e, por isso, pode ser considerada eficaz como modelo de resposta.

Na seção 4.5, encontram-se os detalhes das respostas obtidas durante os testes. Como resultado, foi possível observar a amplitude de possibilidades que esse tipo de funcionalidade oferece. A partir desses testes, destacam-se aplicações como consultas a materiais de disciplinas, criação de questionários baseados em conteúdos fornecidos pelos professores, e a autocorreção de exercícios dos alunos, apresentando porcentagens de acertos e erros. Isto permite que professores melhorem sua compreensão e identifiquem pontos que requeiram reforço, além de maior atenção e estudo por parte dos estudantes. Essas funcionalidades oferecem potencial significativo para a inovação e aprimoramento do processo de ensino e avaliação no ambiente educacional.

5.5 Tabelas Comparativas

Nesta seção, foram criadas duas tabelas a partir dos testes realizados. Elas têm o objetivo de instruir e agregar valor ao usuário, proporcionando uma compreensão mais clara e simplificada dos resultados gerais. A apresentação foi subdividida em duas partes:

Pontuação:

- Corretude: 100% — A lógica está correta, e o código cumpre seu objetivo.
- Melhoria:
 - Clareza: Poderia ser mais explícito nas mensagens de input, como "Digite o primeiro número" e "Digite o segundo número", para melhorar a interação com o usuário.

Recomendações:

- Utilizar mensagens de entrada mais detalhadas, como "Digite o primeiro número", torna o programa mais amigável e autoexplicativo para o usuário.

Exemplo atualizado:

```

python
num1 = float(input("Digite o primeiro número: "))
num2 = float(input("Digite o segundo número: "))
soma = num1 + num2
print("A soma é:", soma)

```

Se precisar de algo mais relacionado a este PDF, é só avisar!

O aluno respondeu da seguinte maneira:

```

python
n1 = float(input("numero 1: "))
n2 = float(input("numero 2: "))
soma = n1 + n2
print("A soma é:", soma)

```

Avaliação:

- Corretude: O código está correto e funcional, calculando a soma de dois números fornecidos pelo usuário e exibindo o resultado. A estrutura básica é a mesma da resposta esperada.
- Diferenças: O único detalhe que difere da resposta esperada é o nome das variáveis (`n1` e `n2` em vez de `num1` e `num2`), o que não afeta o funcionamento do programa. A mensagem de entrada também está levemente diferente ("numero 1" em vez de "Digite o primeiro número").

Figura 29 – Resposta de pergunta do ChatGPT sobre questões em pdf

Definição de Algoritmo

Um algoritmo é um conjunto finito de instruções ou passos bem definidos que são executados em uma sequência específica para resolver um problema ou realizar uma tarefa. Os algoritmos são fundamentais em computação e programação, pois estabelecem uma lógica que os computadores seguem para executar operações. Um bom algoritmo deve ser claro, eficaz e deve ter uma terminologia bem definida.

Algoritmo para Imprimir "Olá, Mundo"

Aqui está um exemplo de algoritmo simples que imprime a mensagem "Olá, mundo" na tela:

1. Iniciar.
2. Definir a mensagem como "Olá, mundo".
3. Exibir a mensagem na tela.
4. Finalizar.

Implementação em Código

Abaixo está a implementação desse algoritmo em algumas linguagens de programação populares:

Python:

```

python
print("Olá, mundo")

```

Java:

```

java
public class Main {
    public static void main(String[] args) {
        System.out.println("Olá, mundo");
    }
}

```

JavaScript:

```

javascript
console.log("Olá, mundo");

```

Cada exemplo de código acima segue o algoritmo definido e exibirá "Olá, mundo" na tela quando executado.

Figura 30 – Resposta de pergunta individual com contexto e configuração personalizada do ChatGPT sobre algoritmo

Tabela 1 – Comparação de suporte a funcionalidades entre ChatGPT, Gemini e Copilot

Funcionalidade	ChatGPT	Gemini	Copilot
Suporte a envio de PDFs	Sim	Não	Não
Multimodal (texto e imagens)	Sim	Sim	Sim
Recomendações de conteúdo adicional (sites/tabelas)	Parcial	Sim	Não
Integração com ferramentas Externas	Não	Sim	Sim
Limite de uso gratuito	Sim	Sim	Sim
Apresentação de referência no final	Não	Não	Sim

a seção 5.5.1, que contém a tabela de suporte a funcionalidades, e a seção 5.5.2, que apresenta a tabela de comparação das ferramentas GenAI e critérios, destacando os principais critérios observados nos testes e associados às três ferramentas.

5.5.1 Limitações

As ferramentas de GenAI, apesar de trazerem grandes benefícios em suas utilizações, muitas vezes possuem limites, e diferentes ferramentas podem não oferecer suporte para certos requisitos específicos. Nesta seção, será abordada a existência ou não de suporte a alguns elementos destacados e percebidos durante os testes.

Para um entendimento mais claro, foram utilizadas três respostas nesta tabela: 'Sim', 'Não' e 'Parcial'. Esses campos foram preenchidos com base em informações geradas a partir dos testes realizados e também de estudos sobre cada ferramenta.

A tabela 1 apresentada faz uma comparação entre as funcionalidades das três ferramentas GenAI: ChatGPT, Gemini e Copilot. Na linha de envio de PDFs, somente o ChatGPT oferece a funcionalidade, enquanto o Gemini e Copilot não a possui. Todos os modelos são multimodais, ou seja, todos podem trabalhar tanto com texto quanto com imagens. No entanto, em relação de recomendar conteúdo adicional, como sugestões de sites e tabelas, o Gemini se destaca por oferecer suporte, enquanto o ChatGPT atende parcialmente e o Copilot não oferece essa funcionalidade. A integração com ferramentas externas está presente no Gemini e no Copilot, mas o ChatGPT não inclui essa opção de forma gratuita, o que traz um ponto negativo para ele. Sobre os níveis gratuitos, todos tem limitações, eles oferecem algum nível de uso gratuito, permitindo o acesso básico sem custo, mas limitam algumas ações. Por fim, apenas o Copilot apresenta referências no final das respostas, isso pode ser um ponto importante em ambientes acadêmicos, enquanto o ChatGPT e o Gemini não incluem essa funcionalidade. Essa análise comparativa pode ajudar na escolha do assistente mais adequado de acordo com as necessidades do envolvido.

5.5.2 Ferramentas GenAI e Critérios

As motivações para a criação da tabela 2 foi apresentar um panorama geral em cima de requisitos desejados e o nível de qualidade das respostas em cada ferramenta,

buscando facilitar a comparação entre elas, com base nos critérios considerados. A forma de preenchimento dos campos da tabela foi baseado nos resultados dos testes, comparando as respostas com cada critério utilizado. Para algumas linhas da tabela, foram necessários a junção de dois ou mais critérios utilizados nos testes, para uma maior exatidão no resultado final.

Para melhor visualização, cinco níveis foram considerados: muito ruim (0-1), ruim (1-2), bom (2-3), muito bom (3-4), excelente (4-5). A média das pontuações dos critérios de cada teste e modelo de teste foi calculada, e esse valor foi associado a um desses rótulos.

Assim as notas foram rotuladas. Pode-se notar que em sua grande maioria se manteve entre muito bom, bom e excelente, validando que as várias funcionalidades são possíveis nas diversas ferramentas, porém algumas funcionam ou respondem de maneiras diferentes, sendo ruim ou excelente conforme motivação desejada.

Para aplicar as notas e os valores das tabelas, foi realizada uma verificação da relação entre os critérios dos testes e suas respectivas pontuações, assim como dos modelos de testes. No caso de critérios de avaliação, como o de 'Perguntas Agrupadas', foi necessária a extração das pontuações de todos os testes agrupados, com e sem contexto, de ambas as rodadas. Esses valores foram somados e, em seguida, calculada a média, resultando em um valor que se enquadrava no intervalo dos rótulos de qualidade. Esse rótulo foi então atribuído à célula da tabela correspondente. Pode-se observar nos resultados da tabela que, na maioria dos casos, as notas se mantiveram entre bom, muito bom e excelente, validando que muitas funcionalidades são possíveis nas diversas ferramentas. No entanto, algumas funcionalidades podem operar ou responder de maneiras distintas, variando de ruim a excelente, dependendo da finalidade desejada.

Tabela 2 – Comparação de desempenho com base nos critérios de avaliação entre ChatGPT, Gemini e Copilot

Critério de Avaliação	ChatGPT	Gemini	Copilot
Respostas objetivas	Excelente	Muito Bom	Bom
Profundidade das respostas	Excelente	Muito Bom	Bom
Proximidade da resposta	Bom	Bom	Bom
Resposta comprehensível	Excelente	Bom	Bom
Explicação conceitual detalhada	Excelente	Bom	Ruim
Utilização de contexto antes da pergunta	Excelente	Excelente	Muito Bom
Perguntas sem texto introdutório de contexto	Muito Bom	Muito Bom	Muito Bom
Resumos do conteúdo respondido	Excelente	Muito Bom	Bom
Capacidade de personalização	Muito Bom	Bom	Muito Bom
Resposta satisfatória	Excelente	Muito Bom	Excelente
Respostas de acordo com a pergunta	Muito Bom	Bom	Bom
Repostas corretas	Excelente	Muito Bom	Muito Bom
Resposta de conceitos básicos	Excelente	Bom	Muito Bom
Resposta de conceitos complexos	Muito Bom	Muito Bom	Bom
Capacidade de exemplificação	Muito Bom	Bom	Muito Bom
Capacidade de análise crítica	Muito Bom	Muito Bom	Bom
Comparação entre tópicos	Muito Bom	Excelente	Bom
Envolvimento de código de programação nas respostas	Muito Bom	Bom	Excelente
Perguntas enviadas individualmente	Excelente	Excelente	Excelente
Perguntas mandadas agrupadas (todas de uma vez)	Muito Bom	Ruim	Bom

6 Conclusão

Nos testes realizados, foi possível observar diferenças significativas no desempenho das três ferramentas de IA (ChatGPT, Gemini e Copilot) em diferentes cenários. Com as 20 perguntas divididas em duas rodadas, ficou claro que enviar as perguntas de forma individual e com contexto proporcionou um melhor resultado em termos de qualidade de resposta. Isso porque as ferramentas puderam focar melhor em cada questão e fornecer respostas mais detalhadas e direcionadas.

O ChatGPT se destacou em explicações detalhadas e claras, sendo excelente para quem precisa de respostas bem explicativas e aprofundadas. Além disso, sua capacidade de aceitar PDFs como entrada mostrou-se uma vantagem quando se busca analisar documentos mais extensos ou complexos. Já o Gemini demonstrou ser muito eficiente em criar tabelas comparativas e resumos dos principais pontos, o que é ótimo para quem procura uma análise rápida e variada, incluindo links úteis para vídeos do YouTube e outros sites informativos. O Copilot, por outro lado, provou ser uma escolha excelente para quem busca suporte direto no desenvolvimento de código. Ele forneceu trechos de código de alta qualidade na maioria das respostas, o que pode ser uma grande ajuda para programadores que buscam soluções rápidas e práticas.

No geral, cada ferramenta mostrou seus pontos fortes, com o ChatGPT se destacando em clareza e profundidade, o Gemini em organização e variação de conteúdo, e o Copilot na entrega de códigos eficazes e precisos. Com base nisso, fica entendido que a escolha da melhor ferramenta depende do tipo de tarefa e do que o usuário está buscando, mas o uso de contexto e a abordagem de envio individual de perguntas são estratégias que claramente elevam a qualidade das respostas fornecidas.

Essas ferramentas são atualizadas periodicamente e estão em constante evolução. Isso significa que testes realizados hoje podem produzir resultados ligeiramente diferentes no futuro. No entanto, a essência e a base das ferramentas permanecem, podendo ser aprimoradas ao longo dos anos com novas funcionalidades e recursos, o que enriquece o conteúdo e oferece uma base sólida para estudos futuros.

No contexto educacional, a utilização de IA representa um grande avanço. Estudos focados nesse nicho são essenciais para a evolução da educação, impactando a forma como os alunos enxergam e interagem com o aprendizado. O uso de GenAI pode potencializar a absorção de conteúdo de maneira mais profunda, podendo contribuir para o desenvolvimento dos alunos.

No campo da programação, o uso de IA é ainda mais promissor, dada a ligação intrínseca entre tecnologia e educação. Essa combinação é valiosa, pois a programação

envolve conteúdos complexos que podem gerar dúvidas e dificuldades. Este estudo poderá servir de base para futuros trabalhos voltados à criação de sistemas tutores inteligentes, que auxiliem alunos e professores na escolha da melhor ferramenta de acordo com a usabilidade. Esses sistemas podem incluir módulos de validação de código de acordo com exercícios propostos e possíveis integrações com ferramentas amplamente adotadas pelas instituições, como o Microsoft Teams, da Microsoft, que também oferece o Copilot, por exemplo. O auxílio de IA voltadas para o ensino de programação é um diferencial significativo e um poderoso catalisador para o crescimento dos alunos.

Referências

ADETAYO, A. J.; ABORISADE, M. O.; SANNI, B. A. Microsoft copilot and anthropic claude ai in education and library service. **Library Hi Tech News**, Emerald Publishing Limited, 2024. Citado na página 14.

ALEXANDER, B.; ASHFORD-ROWE, K.; BARAJAS-MURPHY, N.; DOBBIN, G.; KNOTT, J.; MCCORMACK, M.; WEBER, N. Educause horizon report. **Higher Education Edition**). Louisville, Co: Educause, 2019. Citado na página 7.

AN, J.; DING, W.; LIN, C. Chatgpt. **tackle the growing carbon footprint of generative AI**, v. 615, p. 586, 2023. Citado na página 12.

BAIDOO-ANU, D.; ANSAH, L. O. Education in the era of generative artificial intelligence (ai): Understanding the potential benefits of chatgpt in promoting teaching and learning. **Journal of AI**, Izmir Academy Association, v. 7, n. 1, p. 52–62, 2023. Citado na página 16.

BATT, S. What is microsoft copilot? how to use copilot in windows. **MSN News**, 2023. Available at: <<https://www.msn.com/en-us/news/technology/what-is-microsoft-copilot-how-to-use-copilot-in-windows/ar-AA18TQ6Y>> (accessed 17 July 2024). Citado na página 14.

BROWN, J.; GREEN, K. Adaptive learning technologies for personalized education. **Journal of Educational Technology Research**, v. 34, n. 2, p. 123–140, 2021. Citado na página 11.

BROWN, T. B.; MANN, B.; RYDER, N.; SUBBIAH, M.; KAPLAN, J.; DHARIWAL, P.; NEELAKANTAN, A.; SHYAM, P.; SASTRY, G.; ASKELL, A.; AGARWAL, S.; HERBERT-VOSS, A.; KRUEGER, G.; HENIGHAN, T.; CHILD, R.; RAMESH, A.; ZIEGLER, D. M.; WU, J.; WINTER, C.; HESSE, C.; CHEN, M.; SIGLER, E.; LITWIN, M.; GRAY, S.; CHESS, B.; CLARK, J.; BERNER, C.; MCCANDLISH, S.; RADFORD, A.; SUTSKEVER, I.; AMODEI, D. **Language Models are Few-Shot Learners**. 2020. Disponível em: <<https://arxiv.org/abs/2005.14165>>. Citado na página 12.

CHEN, B.; ZHANG, Z.; LANGRENÉ, N.; ZHU, S. Unleashing the potential of prompt engineering in large language models: a comprehensive review. **arXiv preprint arXiv:2310.14735**, 2023. Citado na página 16.

CHO, K.; MERRIENBOER, B. van; GULCEHRE, C.; BAHDANAU, D.; BOUGARES, F.; SCHWENK, H.; BENGIO, Y. **Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation**. 2014. Disponível em: <<https://arxiv.org/abs/1406.1078>>. Citado na página 13.

COOPER, G. Examining science education in chatgpt: An exploratory study of generative artificial intelligence. **Journal of Science Education and Technology**, Springer, v. 32, n. 3, p. 444–452, 2023. Citado na página 15.

- DAVIS, S.; FREDERICKS, R. Student engagement in computer science education. **Education and Information Technologies**, v. 22, n. 3, p. 989–1006, 2017. Citado na página 11.
- DEVLIN, J.; CHANG, M.-W.; LEE, K.; TOUTANOVA, K. **BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding**. 2019. Disponível em: <<https://arxiv.org/abs/1810.04805>>. Citado na página 12.
- GO, E.; SUNDAR, S. S. Humanizing chatbots: The effects of visual, identity and conversational cues on humanness perceptions. **Computers in human behavior**, Elsevier, v. 97, p. 304–316, 2019. Citado na página 10.
- GOOGLE. **Cotas do Google Gemini**. 2024. Acesso em: 10 out. 2024. Disponível em: <<https://cloud.google.com/gemini/docs/quotas?hl=pt-br>>. Citado na página 13.
- _____. **Limitações e Forças do AI Generativo da Vertex AI**. 2024. Acesso em: 10 out. 2024. Disponível em: <<https://cloud.google.com/vertex-ai/generative-ai/docs/multimodal/strengths-limits?hl=pt-br>>. Citado na página 13.
- _____. **Suporte de Limitações e Cotas de API do Google**. 2024. Acesso em: 10 out. 2024. Disponível em: <<https://support.google.com/a/users/answer/14796585?hl=en>>. Citado na página 13.
- GUPTA, T. **The Role Of Generative AI In Natural Language Processing**. 2024. [Online; accessed 22-marco-2024]. Disponível em: <<https://elearningindustry.com/role-of-generative-ai-in-natural-language-processing>>. Citado na página 11.
- IMRAN, M.; ALMUSHARRAF, N. Google gemini as a next generation ai educational tool: a review of emerging educational technology. **Smart Learning Environments**, Springer, v. 11, n. 1, p. 22, 2024. Citado na página 13.
- JARA, I.; OCHOA, J. M. Usos y efectos de la inteligencia artificial en educación. **Sector Social división educación. Documento para discusión número IDB-DP-00-776. BID**. doi: <http://dx.doi.org/10.18235/0002380>, 2020. Citado na página 7.
- JIANG, Y.; LI, X.; LUO, H.; YIN, S.; KAYNAK, O. Quo vadis artificial intelligence? **Discover Artificial Intelligence**, Springer, v. 2, n. 1, p. 4, 2022. Citado na página 9.
- KOROTEEV, M. Bert: a review of applications in natural language processing and understanding. **arXiv preprint arXiv:2103.11943**, 2021. Citado na página 10.
- KOSTKA, I.; TONCELLI, R. Exploring applications of chatgpt to english language teaching: Opportunities, challenges, and recommendations. **TESL-EJ**, ERIC, v. 27, n. 3, p. n3, 2023. Citado na página 7.
- LEE, S.; KIM, Y. Enhancing programming education through ai-driven learning platforms. **Educational Technology Journal**, v. 42, n. 4, p. 300–315, 2020. Citado na página 11.
- LEITE, B. S. Generative artificial intelligence in chemistry teaching: Chatgpt, gemini, and copilot's content responses. **Journal of Applied Learning and Teaching**, v. 7, n. 2, 2024. Citado na página 15.

- MARTIN, J. H. **Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition.** [S.l.]: Pearson/Prentice Hall, 2009. Citado na página 10.
- MEKNI, M. An artificial intelligence based virtual assistant using conversational agents. **Journal of Software Engineering and Applications**, Scientific Research Publishing, v. 14, n. 9, p. 455–473, 2021. Citado na página 13.
- MICROSOFT. **Comparação entre o Copilot Gratuito e Pago do Microsoft 365.** 2024. Acesso em: 10 out. 2024. Disponível em: <<https://www.geeky-gadgets.com/microsoft-365-copilot-for-free-vs-paid-comparison/?form=MG0AV3>>. Citado na página 14.
- _____. **Microsoft 365 Copilot - Página Oficial.** 2024. Acesso em: 10 out. 2024. Disponível em: <<https://www.microsoft.com/pt-br/microsoft-365/copilot/?form=MG0AV3>>. Citado na página 14.
- _____. **Microsoft Aumenta Limite Diário do Copilot Gratuito.** 2024. Acesso em: 10 out. 2024. Disponível em: <<https://mspoweruser.com/microsoft-lifts-free-copilot-daily-limit-rolling-out-amidst-buzz-over-paid-copilot-pro-plan/?form=MG0AV3>>. Citado na página 14.
- OPENAI. **Central de Ajuda da OpenAI.** 2024. Acesso em: 10 out. 2024. Disponível em: <<https://help.openai.com/en/>>. Citado na página 12.
- _____. **Preços do ChatGPT da OpenAI.** 2024. Acesso em: 10 out. 2024. Disponível em: <<https://openai.com/chatgpt/pricing/>>. Citado na página 12.
- RADFORD, A.; NARASIMHAN, K.; SALIMANS, T.; SUTSKEVER, I. et al. Improving language understanding by generative pre-training. San Francisco, CA, USA, 2018. Citado na página 12.
- REBELO, M. What is microsoft copilot? **Zapier Blog**, December 2023. Available at: <<https://www.zapier.com/blog/microsoft-copilot/>> (accessed 17 July 2024). Citado na página 13.
- REIS, R. L. d. O.; SOUZA, F. Desenvolvimento de um jogo educativo para apoiar o processo de aprendizagem das camadas de redes de computadores. **Revista Científica Multidisciplinar Núcleo do Conhecimento**, v. 05, n. 05, p. 41–72, 2020. Citado na página 42.
- ROBINS, A.; ROUNTREE, J.; ROUNTREE, N. Learning and teaching programming: A review and discussion. **Computer Science Education**, v. 13, n. 2, p. 137–172, 2003. Citado na página 11.
- RUSSELL, S. Artificial intelligence and the problem of control. **Perspectives on Digital Humanism**, v. 19, 2022. Citado na página 9.
- RUSSELL, S.; NORVIG, P. **Artificial Intelligence: A Modern Approach.** 3rd. ed. [S.l.]: Prentice Hall, 2016. Citado na página 9.
- RUSSELL, S. J.; NORVIG, P. **Artificial intelligence: a modern approach.** [S.l.]: Pearson, 2016. Citado na página 10.

SAEIDNIA, H. R. Welcome to the gemini era: Google deepmind and the information industry. **Library Hi Tech News**, Emerald Publishing Limited, n. ahead-of-print, 2023. Citado na página 13.

SCHACHNER, T.; KELLER, R.; WANGENHEIM, F. v. Artificial intelligence-based conversational agents for chronic conditions: systematic literature review. **Journal of medical Internet research**, JMIR Publications Toronto, Canada, v. 22, n. 9, p. e20701, 2020. Citado na página 11.

SMITH, A.; JONES, B. Ensuring quality and accessibility in programming education resources. **Journal of Educational Technology Systems**, v. 36, n. 1, p. 45–60, 2019. Citado na página 11.

VASWANI, A.; SHAZEE, N.; PARMAR, N.; USZKOREIT, J.; JONES, L.; GOMEZ, A. N.; KAISER, L.; POLOSUKHIN, I. **Attention Is All You Need**. 2023. Disponível em: <<https://arxiv.org/abs/1706.03762>>. Citado na página 12.

WANG, Q.; LIU, S. Data analytics in programming education: Opportunities and challenges. **Journal of Educational Technology Systems**, v. 36, n. 3, p. 201–215, 2020. Citado na página 11.

WIKIPÉDIA. **ChatGPT — Wikipédia, a encyclopédia livre**. 2024. [Online; accessed 9-fevereiro-2024]. Disponível em: <<https://pt.wikipedia.org/w/index.php?title=ChatGPT&oldid=67453136>>. Citado 2 vezes nas páginas 7 e 12.

Apêndices

APÊNDICE A – Perguntas da Primeira e Segunda Rodada de Testes

Segue em apêndice a descrição das 30 perguntas totais das duas rodadas de testes nas três ferramentas de GenAI (ChatGPT, Gemini e Copilot). Este capítulo está separado por etapa de teste, na seção 2.4 traz as perguntas da primeira rodada, enquanto as 2.5 as perguntas da segunda rodada. O conteúdo dos exercícios foram segregadas por temas e fornecidas em lista.

A.1 Primeira Rodada

1. Tema: Programação

- Pergunta 1: O que é uma variável e quais são os tipos de variáveis mais comuns em programação?
- Pergunta 2: Explique a diferença entre uma função e um método.

2. Tema: Estruturas de Dados

- Pergunta 1: O que é um array e como ele difere de uma lista ligada?
- Pergunta 2: Como funciona uma pilha (stack) e uma fila (queue)?

3. Tema: Algoritmos

- Pergunta 1: O que é a complexidade de tempo de um algoritmo e por que é importante?
- Pergunta 2: Explique a diferença entre busca linear e busca binária.

4. Tema: Sistemas Operacionais

- Pergunta 1: O que é um processo e como ele difere de uma thread?
- Pergunta 2: O que é gerenciamento de memória em sistemas operacionais?

5. Tema: Banco de Dados

- Pergunta 1: O que é uma chave primária em um banco de dados relacional?
- Pergunta 2: Explique a diferença entre SQL e NoSQL.

6. Tema: Redes de Computadores

- Pergunta 1: O que é o protocolo TCP/IP?
- Pergunta 2: Qual é a diferença entre um switch e um roteador?

7. Tema: Engenharia de Software

- Pergunta 1: O que é o ciclo de vida de desenvolvimento de software (SDLC)?
- Pergunta 2: Explique a diferença entre metodologias ágeis e tradicionais (como o modelo Waterfall).

8. Tema: Inteligência Artificial

- Pergunta 1: O que é aprendizado supervisionado e não supervisionado em machine learning?
- Pergunta 2: Explique o que é um algoritmo de busca A*.

9. Tema: Segurança da Informação

- Pergunta 1: O que é criptografia e qual a diferença entre criptografia simétrica e assimétrica?
- Pergunta 2: O que é um firewall e qual sua função principal em uma rede de computadores?

10. Tema: Teoria da Computação

- Pergunta 1: O que é uma máquina de Turing?
- Pergunta 2: Explique a diferença entre problemas P e NP.

A.2 Segunda Rodada

1. Tema: Imprimindo uma Mensagem

- Pergunta: Faça uma definição do que é algoritmo e depois escreva um para impressão de uma mensagem "olá, mundo" na tela.

2. Tema: Variáveis e Operações Aritméticas

- Pergunta: Escreva um programa que peça ao usuário dois números e exiba a soma deles depois mostre um exemplo de execução.

3. Tema: Estruturas Condicionais

- Pergunta: Escreva um programa que peça um número ao usuário e informe se o número é par ou ímpar e mostre um exemplo de execução.

4. Tema: Laços de Repetição

- Pergunta: Explique o que acontece quando um laço de repetição entra em loop e escreva um programa que exiba os números de 1 a 10 usando um laço for e exemplos de como entrar em loop, mostre um exemplo de execução.

5. Tema: Listas

- Pergunta: Escreva um programa que peça ao usuário 5 números, armazene-os em uma lista e, em seguida, exiba a lista na tela, depois explique cada linha e mostre um exemplo de execução.

6. Tema: Funções

- Pergunta: Explique como posso construir minhas próprias funções e em seguida escreva uma função chamada fatorial que receba um número e retorne o fatorial desse número, mostre um exemplo de execução.

7. Tema: Manipulação de Strings

- Pergunta: Informe as diferenças de como são tratadas as strings nas linguagens de programação c e python, depois escreva um programa que conte o número de vogais em uma string fornecida pelo usuário e mostre um exemplo de execução.

8. Tema: Dicionários

- Pergunta: Escreva um programa que leia uma frase do usuário e conte a frequência de cada palavra na frase, exibindo os resultados em um dicionário, mostre um exemplo de execução e explique o que acontece se tentar acessar um termo que a chave não existe.

9. Tema: Classes e Objetos

- Pergunta: Crie uma classe chamada Pessoa que tenha os atributos nome e idade. Adicione um método de aniversário que aumenta a idade da pessoa em 1. Crie uma instância dessa classe e demonstre o funcionamento do método aniversário, mostre um exemplo de execução.

10. Tema: Algoritmos de Ordenação

- Pergunta: Implemente o algoritmo de ordenação Merge Sort para ordenar uma lista de números fornecida pelo usuário e mostre um exemplo de execução.